Lübbes Computerbücher

HOWARD BUDIN

Computerspiele mit TRS-80

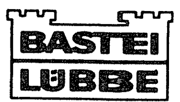


BASTEI LÜBBE

HOWARD BUDIN

Computerspiele mit TRS-80

Aus dem Amerikanischen übersetzt
von Bernd Gatter



BASTEI-LÜBBE-TASCHENBUCH
Band 63 087

Deutsche Erstveröffentlichung

Titel der Originalausgabe:
SPEED WALKER — FUN TO PROGRAM YOUR TRS-80

Erschienen bei Pinnacle Books, Inc., New York

Printed in Western Germany 1986

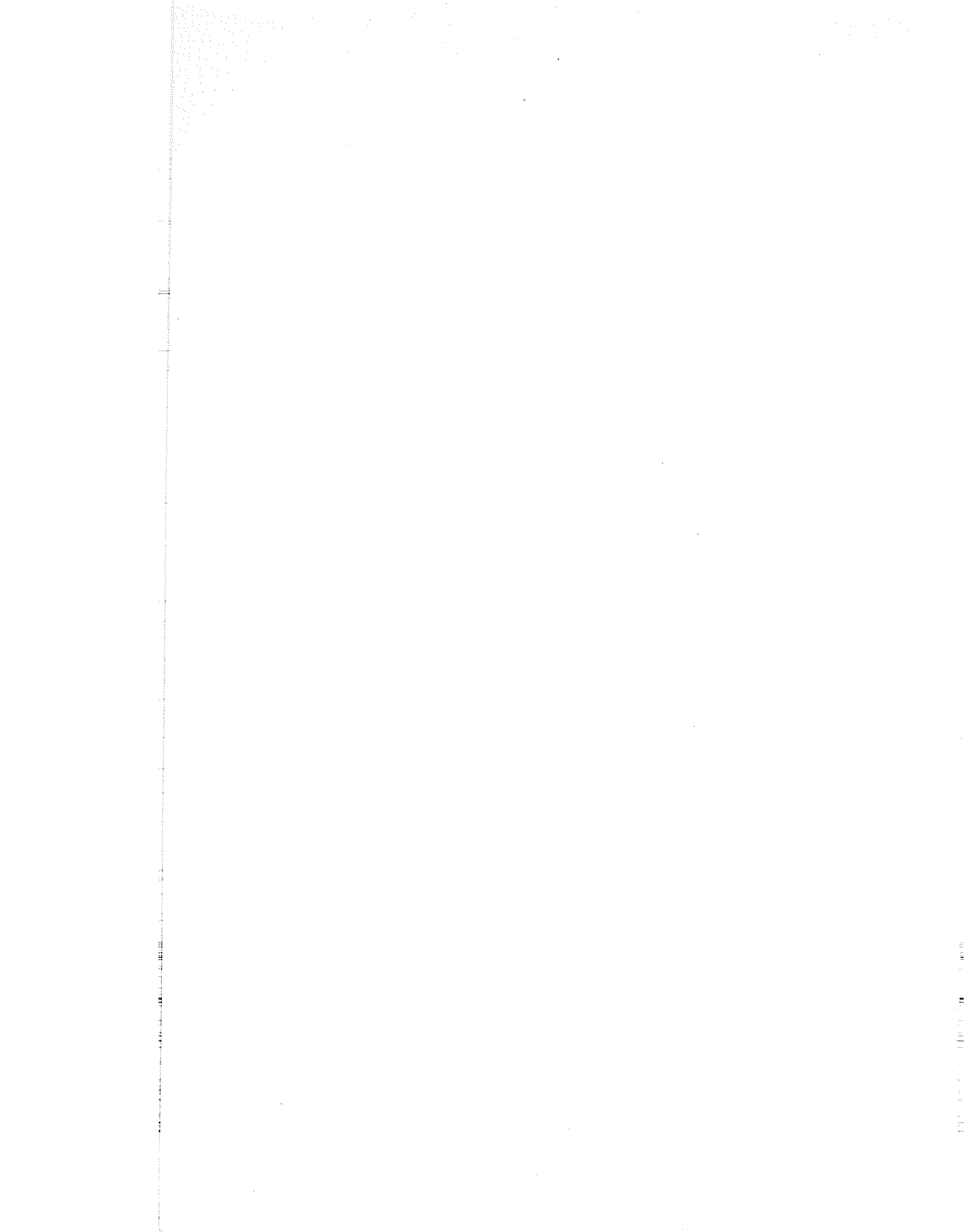
Einbandgestaltung: Roberto Patelli

Illustrationen: Cris Hammond
Satz: Fotosatz Böhm, Köln
Gesamtherstellung: Ebner Ulm

ISBN 3-404-63087-4

Der Preis dieses Bandes versteht sich einschließlich der gesetzlichen Mehrwertsteuer.

Inhalt



Einleitung

Willkommen bei der Programmierung mit Speed Walker! Dieses Buch wurde für all diejenigen geschrieben, die schon etwas von BASIC verstehen, und es anwenden wollen, um ein paar interessante Programme herzustellen. Wir nehmen an, daß Sie die folgenden BASIC-Kommandos kennen:

REM	FOR/NEXT	END
PRINT	IF/THEN	
INPUT	GOTO	

Machen Sie sich keine Gedanken, wenn Sie kein Experte in der Benutzung mancher Programmbefehle sind — es wird nicht lange dauern, und Sie werden meisterhaft damit umgehen können. Ein Programmierer sein heißt ein Detektiv sein — Sie müssen ständig neue Geheimnisse in Programmen aufdecken. Programmierer verbringen die Hälfte ihres Lebens mit der Suche nach möglichen Fehlerquellen in ihren Programmen. Um Ihnen nun das Lesen, Verändern und Testen Ihrer Programme leichter zu machen, benutzen wir in diesem Buch die strukturierte Programmierung. Das bedeutet, daß wir zunächst die Hauptteile des Programms planen und dann erst in die Her-

stellung der Einzelteile einsteigen. Wir benutzen viele REM-Anweisungen, um die Bedeutung von Variablen, die wir verwenden, und den Inhalt von einzelnen Programmteilen genau zu dokumentieren. Wir rücken auch einzelne Programmbausteine ein, um sie von anderen abzuheben beziehungsweise ihre Abhängigkeit von anderen Befehlen aufzuzeigen.

Wenn wir uns auf dieses Spiel einlassen, werden wir mysteriöse Techniken kennenlernen: die Voraussage der Zukunft, die Verschlüsselung und Dechiffrierung von geheimen Nachrichten, die Bewegung von Objekten kreuz und quer über den ganzen Schirm. In jedem Kapitel bauen wir neue Konzepte auf, erweitert sich unser Befehlsvorrat, und die Kombination schon bekannter und neuer Techniken führt dann zu einem weiteren Spiel. Jeder neue Gedanke wird eingehend behandelt. Darüber hinaus gibt es dazu Beispiele, die Sie in Ihren Computer eingeben und ausprobieren können. Am Ende des letzten Kapitels haben wir dann ein weitaus komplizierteres Programm konstruiert als das, mit dem wir im ersten Kapitel begonnen haben. Aber im Laufe der Zeit ist das auch kein Problem, da wir uns Zug um Zug mit allen Bausteinen dieses Programms anfreunden werden.

Am Ende eines jeden Kapitels bieten wir Ihnen noch verschiedene Variationsmöglichkeiten an, die das gerade aufgebaute Programm vielleicht noch interessanter machen. In gewissem Sinne werden nämlich Programme nie fertig — Sie können immer noch die eine oder andere Ergänzung

oder Verbesserung hinzufügen. Und das ist das Schöne an der Programmierung: Sie können Ihre eigenen Ideen einbringen. Obwohl wir in allen Kapiteln »schlüsselfertige« Programme darstellen, die Sie eingeben und sofort durchführen können, ist das Ende dieser Programme wirklich noch offen. Wir wollen sogar, daß Sie sie verändern und verbessern, daß Sie neue Elemente einfügen und somit Ihre eigenen Programme schreiben. Aber vor allem wollen wir, daß Sie Spaß an der Programmierung und an diesem Buch finden.

1
Wie steht's mit Ihrer Zukunft?

Stellen Sie sich einmal vor, Sie sitzen in einem chinesischen Restaurant und öffnen eins der chinesischen Glückskekse. Haben Sie sich je gefragt, wie gerade diese spezielle Zukunftsvoraussage in Ihr Plätzchen kommt? Ob es wohl jemanden gibt, der wollte, daß Sie genau dieses Plätzchen bekommen? Vermutlich nicht. Irgend jemand muß sich wohl hingesetzt und eine Menge solcher Vorhersagen niedergeschrieben haben, ein anderer hat sie dann alle in die kleinen Kuchen gesteckt. In jedem Fall handelt es sich hier aber um eine »Zufallsarbeit«. Mit anderen Worten: Sie hätten jede beliebige Voraussage, die geschrieben wurde, erhalten können.



Computer können eine ganze Menge, unter anderem können sie auch die Zukunft vorhersagen, und das ohne große Zaubertricks.
Sie wissen schon, daß irgend jemand sie programmieren — ihnen genau sagen muß, welche Schritte sie durchführen sollen. Wir werden nun ein »Blick-in-die-Zukunft-Programm« schreiben. Wir machen das in der gleichen Schrittfolge, als ob wir Weissagungszettel für Glücksplätzchen erstellen würden:

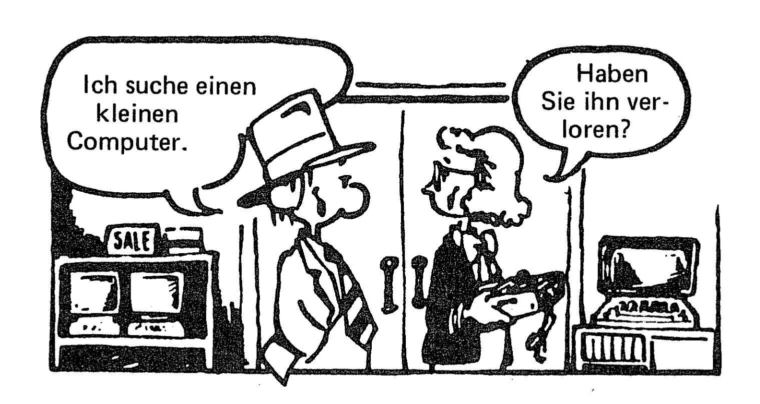
1. Eine Menge Vorhersagen festlegen und speichern,
2. per Zufall eine der Vorhersagen herausfischen,
3. die Zukunft auf dem Bildschirm anzeigen

In diesem Buch werden wir die Programme strukturieren, damit sie leicht zu lesen und zu verändern sind. Wir tun das, indem wir das Programm in ähnliche Schritte unterteilen wie die drei oben genannten. Diese Schritte werden UNTERROUTINEN genannt. Der STEUERUNGSTEIL eines Programms zeigt dem Computer, wo er diese Unterroutinen zu finden hat. Der Steuerungsteil liest sich, als ob er ein Inhaltsverzeichnis des Programms wäre.
Wir benutzen auch eine größere Anzahl von REMarks (Bemerkungszeilen), die uns mitteilen, was jeder Programmbaustein im einzelnen für eine Aufgabe hat: welchen Inhalt die von uns benutzten Variablen haben und welche Schritte innerhalb einer Unterroutine durchgeführt werden. Und so sieht der Steuerungsteil für unser Weissagungsprogramm aus:

```
10 REM——WIE SIEHT IHRE ZUKUNFT
   AUS?————
20 REM F$(4): BEREICH FÜR VIER VOR-
   HERSAGEN
30 REM J      : ZÄHLER
40 REM N      : ZUFALLSZAHL
50 REM——————————
100 GOSUB 1000:REM——SPEICHERUNG
    DER VORHERSAGEN
200 GOSUB 2000:REM——AUSWAHL EI-
    NER VORHERSAGE
300 GOSUB 3000:REM——AUSGABE DER
    VORHERSAGE
400 END
```

Die erste REMarkzeile enthält den Namen des Programms. Die nächsten drei Zeilen beschreiben alle Variablen, die in diesem Programm gebraucht werden. Die Zeilen 100, 200 und 300 zeigen uns und dem Computer, wo sich die Unterroutinen befinden. GOSUB 1000 weist den Computer an, zur Zeile 1000 zu springen und ab da alle Befehle auszuführen bis zum Kommando RETURN, das den Rücksprung zu der Programmstelle bedeutet, die mit GOSUB verlassen wurde.

Die Vorhersagen in einem Bereich speichern



Nun beginnen wir mit dem Schreiben der Unterroutinen. Die auf Zeile 1000 soll so viele Vorhersagen, wie wir wollen, mittels DATA-Anweisungen speichern. Und so sieht der Beginn der Unterroutine aus:

```
1000 REM------SPEICHERUNG DER VOR-
     HERSAGEN------
1010 DATA "SIE WERDEN SEHR REICH
     SEIN"
1020 DATA "SIE WERDEN ÜBERAUS
     GLÜCKLICH SEIN"
1030 DATA "SIE WERDEN EIN COMPUTER-
     PROGRAMMIERER"
1040 DATA "SIE WERDEN AUF EINE LANGE
     REISE GEHEN"
```

Das sind natürlich nur Beispiele. Sie sollten Ihre eigenen Wunschträume eingeben. Achten Sie nur darauf, daß jede Zeile mit dem Wort DATA beginnt und die Weissagungstexte in Anführungsstriche gesetzt werden müssen. DATA-Anweisungen können überall im Programm vorkommen. Wenn der Computer einen READ-Befehl findet, sucht er automatisch nach den zu lesenden Daten. Er beginnt die Suche am Programmanfang und sucht so lange, bis er welche gefunden hat. Bei dem nächsten READ-Befehl, den er erhält, setzt er die Suche hinter den zuletzt gefundenen Daten fort. Da wir in unserem Beispiel vier Datendefinitionen haben, benötigen wir auch vier READ-Befehle:

```
1050 DIM F$(4)
1060 FOR J = 1 TO 4
1070 READ F$(J)
1080 NEXT J
1090 RETURN
```

Die Zeile 1050 markiert für den Computer, daß es sich bei F$ um mehr als eine Variable handelt — die Zahl in Klammern gibt die Anzahl der Datenelemente oder -zellen an. Die Gruppe der vier Vorhersagen wird Bereich oder Tabelle genannt, das sind mehrere Dateneinheiten, für die der gleiche Variablenname verwendet wird.
Die Schleife in den Zeilen 1060 und 1080 befiehlt dem Computer, vier Dateneinheiten (DATA) zu lesen und in F$ zu speichern (READ in 1070): die erste Zeichenkette (Vorhersage) wird mit F$(1),

die zweite mit F$(2) und so weiter bezeichnet. Die Zeile 1090 bringt den Computer dazu, in den Steuerungsteil, den man auch als HAUPTPROGRAMM bezeichnet, zurückzukehren und die Arbeit an der Stelle fortzusetzen, wo er aufgehört hat. Er wird nun die Zeile 200 durchführen, die ihn anweist, in die Zeile 2000 zu verzweigen und eine der Weissagungen auszuwählen.

Eine Vorhersage per Zufall auswählen



Nachdem wir nun vier Zukunftsvisionen als F$(1), F$(2), F$(3) und F$(4) gespeichert haben, müssen wir eine nach dem Zufallsprinzip auswählen. Der TRS-80 ist mit einer eingebauten

Funktion ausgerüstet, die jedesmal, wenn Sie sie benutzen, eine andere Zahl auswählen kann. Wenn wir diese Funktion anweisen, immer eine der Zahlen 1, 2, 3 oder 4 zu nehmen, dann können wir diese Zahl zur Auswahl einer unserer vier F$ genannten Datenzellen benutzen. Da diese Funktion, die übrigens RANDOM-Funktion heißt, so häufig in diesem Buch vorkommt, wollen wir nun besser erst einmal etwas Zeit darauf verwenden, sie kennenzulernen und zu verstehen, wie sie arbeitet.
Bauen wir zunächst einmal ein kleines Testprogramm auf. Wenn Sie schon mit der Eingabe des vorausschauenden Programms begonnen haben, dann sichern Sie bitte die Eingabe mit SAVE und geben das NEW-Kommando ein. Anschließend machen Sie bitte folgende Eingabe:

```
10 FOR X = 1 TO 10
20       PRINT RND(8)
30 NEXT X
```

Das ist schon das gesamte Programm. Führen Sie das Programm mit RUN mehrmals durch, und schauen Sie sich die ausgegebenen Zahlen genau an. Sie erhalten immer wieder eine Zahl zwischen 1 und 8. Nun ändern Sie bitte Zeile 20:

```
20 PRINT RND(25)
```

Bei der anschließenden mehrmaligen Durchführung zeigt Ihnen der Computer immer wieder Zahlen von 1 bis 25. Die RND-Funktion besteht

aus dem Kommando RND und einer in Klammern gefaßten Ganzzahl. Ist diese Ganzzahl 1 oder höher, gibt die RND-Funktion immer einen Wert zwischen 1 und der angegebenen Zahl aus. Wenn Sie eine Zufallszahl zwischen 1 und 50 haben wollen, müssen Sie 50 eingeben. Soll die Ausgabe aus einer Zahl zwischen 1 und 75 bestehen, funktioniert das mit dem Befehl RND(75). Diese Formel inbegriffen, besteht das Unterprogramm aus drei Zeilen:

```
2000 REM------EINE ZUFALLSZAHL GE-
     NERIEREN-----
2010 N = RND(4)
2020 RETURN
```

Wir erhalten nun eine Zufallszahl von 1 bis 4 und speichern sie als Variable N. Nun müssen wir nur noch das Orakel mit der Zahl N auf dem Bildschirm anzeigen.

Die Ausgabe der Vorhersage



Eigentlich sollten Sie keine Probleme mit dem Verstehen der Funktionsweise dieser Unterroutine haben:

```
3000 REM———DIE VORHERSAGE AUSGE-
     BEN——————
3010 CLS:PRINT
3020 PRINT "UND HIER IST IHRE ZU-
     KUNFT:"
3030 PRINT:PRINT
3040 PRINT F$(N)
3050 RETURN
```

Das Kommando CLS löscht den gesamten Bildschirm und positioniert den Cursor auf die erste Stelle der ersten Bildschirmzeile, also links oben. Der folgende PRINT-Befehl ohne Zusatz läßt eine

Zeile leer. Also löscht Zeile 3010 alles, was sich auf dem Schirm befand, und setzt den Cursor auf die zweite Zeile, wo die Meldung »UND HIER IST IHRE ZUKUNFT:« angezeigt wird. Zeile 3030 läßt weitere zwei Zeilen frei, so daß die Prophezeiung N auf der Bildschirmzeile 5 sichtbar wird und die Ausgabe in Zeile 3040 erfolgt. Wie lautet die Prophezeiung N? Wenn N den Wert 1 enthält, wird der Computer F$(1), »SIE WERDEN SEHR REICH SEIN« anzeigen. Enthält N den Wert 3, dann wird F$(3) gezeigt, »SIE WERDEN EIN COMPUTER-PROGRAMMIERER«.

Das Programm zusammenfügen



Alle Unterroutinen sind geschrieben, und das Programm ist bereit zur Durchführung. Auch wenn Sie das ganze Programm durch Nach-

schauen an den verschiedenen Stellen in diesem Kapitel nachsehen können, halten wir es für nützlich, es hier noch einmal in seiner Gesamtheit darzustellen:

```
  10 REM——WIE SIEHT IHRE ZUKUNFT
     AUS?————
  20 REM F$(4): BEREICH FÜR VIER VOR-
     HERSAGEN
  30 REM J      : ZÄHLER
  40 REM N      : ZUFALLSZAHL
  50 REM——————
 100 GOSUB 1000:REM——SPEICHERUNG
     DER VORHERSAGEN
 200 GOSUB 2000:REM——AUSWAHL EI-
     NER VORHERSAGE
 300 GOSUB 3000:REM——AUSGABE DER
     VORHERSAGE
 400 END
1000 REM—————SPEICHERUNG DER VOR-
     HERSAGEN————
1010 DATA "SIE WERDEN SEHR REICH
     SEIN"
1020 DATA "SIE WERDEN ÜBERAUS
     GLÜCKLICH SEIN"
1030 DATA "SIE WERDEN EIN COMPUTER-
     PROGRAMMIERER"
1040 DATA "SIE WERDEN AUF EINE LANGE
     REISE GEHEN"
1050 DIM F$(4)
1060 FOR J = 1 TO 4
1070      READ F$(J)
1080 NEXT J
```

```
1090 RETURN
2000 REM———EINE ZUFALLSZAHL GENE-
     RIEREN——————
2010 N = RND(4)
2020 RETURN
3000 REM————DIE VORHERSAGE AUS-
     GEBEN——————
3010 CLS:PRINT
3020 PRINT "UND HIER IST IHRE ZUKUNFT:"
3030 PRINT:PRINT
3040 PRINT F$(N)
3050 RETURN
```



Variationen



Es gibt viele Dinge, die man nach der Erarbeitung eines Programms noch zu dessen Verschönerung und Abrundung tun kann. Am Ende eines jeden Kapitels wollen wir Ihnen einige Variationen, die Sie ausprobieren können, vorstellen und Tips für deren Realisierung geben. Zunächst einige Variationen für unser Weissagungsprogramm:

1. Vier Voraussagen mögen vielen von Ihnen nicht genug erscheinen. Sie können eigentlich so viele, wie Sie wollen, erfinden und speichern. Natürlich müssen Sie dann einige Teile im Programm anpassen beziehungsweise hinzufügen:
a) Ergänzen Sie die DATA-Zeilen in der ersten Unterroutine.
b) Ändern Sie die Zahl in den Zeilen 1050 und 1060.

c) Ändern Sie auch die 4 in der Random-Funktionszeile 2010.

2. Geben Sie zwei oder mehr Vorhersagen aus: Nehmen Sie an, zwei Menschen wollen ihr Schicksal zur gleichen Zeit erfahren. Sie müssen also zwei Prophezeiungen gleichzeitig auswählen — nennen wir sie N1 und N2:

```
2010 N1 = RND(4)
2015 N2 = RND(4)
```

In der letzten Unterroutine würden Sie in diesem Fall sowohl den Inhalt von F$(N1) als auch von F$(N2) auf dem Bildschirm anzeigen. Sie können natürlich auch durch freizügige Anwendung des PRINT-Befehls den Bildschirm effektvoller gestalten (zum Beispiel durch einen Rahmen, der aus Sternchen besteht). Das sieht dann ungefähr so aus:

Ihr persönliches Horoskop...

Hier sind der Phantasie keine Grenzen — außer denen des Bildschirms — gesetzt.
Auf diese Weise können Sie nun jede beliebige Anzahl an Weissagungen speichern und ausgeben.

2
Sprechen Sie eine Geheimsprache?

Haben Sie jemals einen Geheimcode benutzt, um einem Freund etwas mitzuteilen? Bei allen Codes müssen ganz bestimmte Regeln befolgt werden — wenn Sie diese kennen, können Sie jeden Code problemlos dechiffrieren. Computer können sehr leicht verschlüsseln und entschlüsseln, wenn Sie die Regeln programmieren. In diesem Kapitel werden wir ein Spielprogramm für zwei Personen schreiben. Die erste gibt eine Nachricht ein. Der TRS-80 nimmt die Nachricht auf, setzt sie in einen Code um und zeigt sie verschlüsselt dem zweiten Spieler. Dieser muß dann versuchen, die Nachricht zu entziffern. Es gibt eine große Anzahl unterschiedlicher Codierungen, die wir benutzen können. In diesem Fall werden wir den Umkehrungscode verwenden: der Computer erhält die Nachricht und gibt sie rückwärts wieder aus. Der zweite Spieler muß also beim Entziffern von rechts nach links lesen:

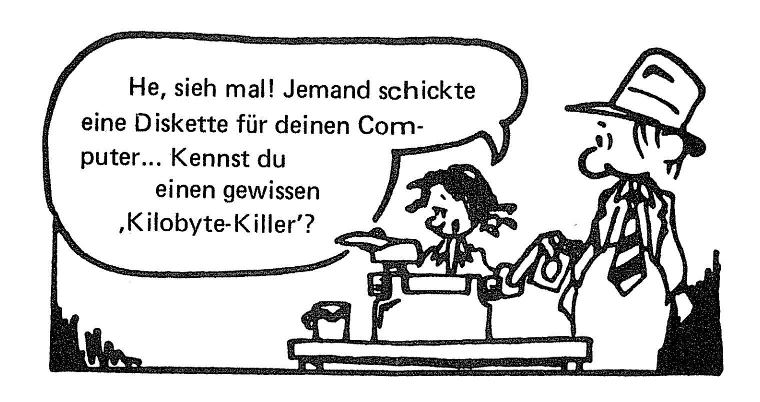
?NESEL SAD EIS NENNÖK

Wenn Sie es können, dann sind Sie in der glücklichen Lage, alles in diesem Programm entschlüsseln zu können:

```
10 REM———————GEHEIMSPRACHE———
   ——
20 REM M$   : DIE NACHRICHT
30 REM L$   : EIN BUCHSTABE IN DER
   NACHRICHT
40 REM C$   : DER SCHLÜSSEL
50 REM A$   : DIE ANTWORT
60 REM J    : EIN ZÄHLER
70 REM———————
100 GOSUB 1000:REM—————DIE NACH-
    RICHT EMPFANGEN
200 GOSUB 2000:REM—————DIE NACH-
    RICHT VERSCHLÜSSELN
300 GOSUB 3000:REM—————QUIZ
400 END
```

Das Hauptprogramm (Steuerungsteil) zeigt uns fünf benötigte Variablennamen und drei Abteilungen (SECTIONS) des Programms in Form von Unterroutinen.

Die Nachricht empfangen



In der Unterroutine auf Zeile 1000 bitten wir darum, daß der zweite Spieler während der Eingabe der Nachricht durch den ersten Spieler nicht zusieht. Die Nachricht wird als Variable M$ gespeichert.

```
1000 REM————DIE NACHRICHT EMP-
     FANGEN————
1010 CLS:PRINT
1020 PRINT "SAGEN SIE IHREM MITSPIE-
     LER, ER SOLL"
1030 PRINT "WEGSEHEN, WENN SIE DIE
     NACHRICHT"
1035 PRINT "EINGEBEN"
1040 PRINT:PRINT
1050 PRINT "GEBEN SIE NUN DIE NACH-
     RICHT EIN"
```

```
1055 PRINT "DANN DRÜCKEN SIE DIE RE-
     TURN-TASTE"
1060 INPUT M$
1070 RETURN
```

Die Nachricht umkehren



Zum Programm gehört eine Zeichenkette, M$ genannt, in ihr kann jedes Zeichen, jeder beliebige Buchstabe oder jede willkürlich eingegebene Ziffer, jedes Sonderzeichen wie beispielsweise das $-Zeichen, Ausrufezeichen, Komma und so weiter enthalten sein. Wir müssen nun die Reihenfolge all dieser Zeichen umkehren und die geänderte Zeichenkette in einer neuen Variablen, die wir C$ nennen werden, abspeichern. Diese Prozedur läuft folgendermaßen ab:

1. Nimm das letzte Zeichen aus M$ und speichere es als erstes Zeichen in C$.
2. Nimm das vorvorletzte Zeichen aus M$ als zweites Zeichen in C$.
3. Nimm das vorvorletzte Zeichen aus M$ als drittes Zeichen in C$.
4. Führe diese Arbeit aus bis zum letzten Zeichen von M$, das das erste Zeichen in C$ werden soll.
Nehmen wir an, daß in M$ das Wort »HALLO« als Nachricht enthalten ist. Der letzte Buchstabe in M$ ist also »O«, und dieser wird der erste Buchstabe in C$. Dann wird das letzte »L« in M$ der zweite Buchstabe in C$. Wenn die Codierung erfolgreich abgeschlossen ist, enthält C$ das Wort

"OLLAH"

Der TRS-80 besitzt eine integrierte Funktion, die jedes gewünschte Zeichen aus einer Zeichenkette auswählen kann. Wir probieren diese spezielle Funktion einmal aus, um ihre Arbeitsweise zu begreifen. Zuerst geben wir ein:

M$="PFERD"

und dann erfolgt die Anweisung:

PRINT MID$(M$,4,1)

(Vergessen Sie aber bitte nicht, die RETURN-Taste zu drücken.)
Der TRS-80 sollte mit dem Buchstaben »R« ant-

worten, da R der vierte Buchstabe in dem in M$ gespeicherten Wort ist. Nun versuchen Sie es mit dem folgenden Kommando:

```
PRINT MID$(M$,3,2)
```

Der TRS-80 wird Ihnen daraufhin zwei Buchstaben aus M$ zeigen, beginnend mit dem dritten Buchstaben des Wortes. Und zwar werden die Buchstaben ER angezeigt, wenn Sie die RETURN-Taste gedrückt haben. Bei der Benutzung der MID$-Funktion werden innerhalb der Klammern drei wesentliche Informationen festgelegt:
Der Name der Zeichenkette.
Die Zeichenposition, mit der begonnen werden soll.
Wie viele Zeichen verarbeitet werden sollen.
Anstelle der direkten Zahlenangaben können auch Variablen genommen werden. Geben Sie beispielsweise dem TRS-80 ein:

```
J=5
```

und anschließend

```
PRINT MID$(M$,J,1)
```

Mit anderen Worten: Sie wollen 1 Zeichen aus M$ sehen, beginnend mit dem Zeichen mit der Nummer J oder 5. In unserem Beispiel ist das der Buchstabe »D«. Was passiert, wenn J den Wert 4 enthält? Dann entspricht MID$(M$,J,1) dem Buchstaben »R«. Durch die Veränderung des

Wertes von J können wir also alle Zeichen in M$ betrachten.
Aber woher kennen wir die Anzahl der Zeichen in M$?
Dies teilt uns eine andere TRS-80-Funktion mit. Versuchen Sie es einmal:

```
PRINT LEN(M$)
```

LEN steht für die Länge irgendeiner Zeichenkette, deren Namen man innerhalb der Klammern angibt. Verändern wir einmal M$ wie folgt:

```
M$="HALLO, SIE DA!"
PRINT LEN(M$)
```

Wenn Sie diese Befehle eingeben, sehen Sie nach dem zweiten Drücken der RETURN-Taste, daß die Länge von M$ dem Wert 13 entspricht. Leerstellen und Satzzeichen zählen natürlich mit, da sie auch Bestandteil der Zeichenkette sind.
Bevor wir beginnen, unsere Unterroutine zu schreiben, probieren wir diese beiden neuen Funktionen in einem kurzen Testprogramm aus, so wie wir es auch in Kapitel 1 mit der RANDOM-Funktion getan haben. Die beste Art und Weise, eine neue Computerlogik zu begreifen, ist, sie im einfachsten Programm, das man sich vorstellen kann, anzuwenden.
Wenn Sie schon einen Teil unseres Programms eingegeben haben, sichern Sie es bitte und geben die folgenden Testanweisungen ein:

```
10 M$ = "HILFE"
20 FOR J = 1 TO 5
30      PRINT MID$(M$,J,1)
40 NEXT J
```

Führen Sie dieses Programm mit RUN durch — die Ausgabe sollte aus den fünf Buchstaben des Wortes HILFE bestehen, die einzeln pro Bildschirmzeile angezeigt werden. Bei der ersten Schleifendurchführung hatte J den Inhalt 1, also wurde der erste Buchstabe ausgegeben. Beim zweiten Mal wurde der zweite Buchstabe gezeigt und so weiter. Nun ändern Sie bitte:

```
10 FOR J = 5 TO 1 STEP —1
```

Führen Sie das Programm erneut durch. Diesmal werden die Buchstaben in umgekehrter Reihenfolge angezeigt. Bei der ersten Schleife trägt J den Wert 5, also wird der fünfte Buchstabe sichtbar. Da J bei jeder Schleife um 1 vermindert wird, entspricht die Variable beim zweiten Mal dem Wert 4, und der vierte Buchstabe wird ausgegeben. Nun modifizieren wir das Testprogramm noch einmal:

```
10 FOR J = LEN(M$) TO 1 STEP —1
```

Das Programm sollte nun völlig gleich ablaufen. Der Computer hat nur den Wert 5 durch LEN(M$) ersetzt, da insgesamt fünf Zeichen in M$ enthalten sind. Weil das Gerät dazu in der Lage ist, müssen wir also die Länge einer als Variable M$ ge-

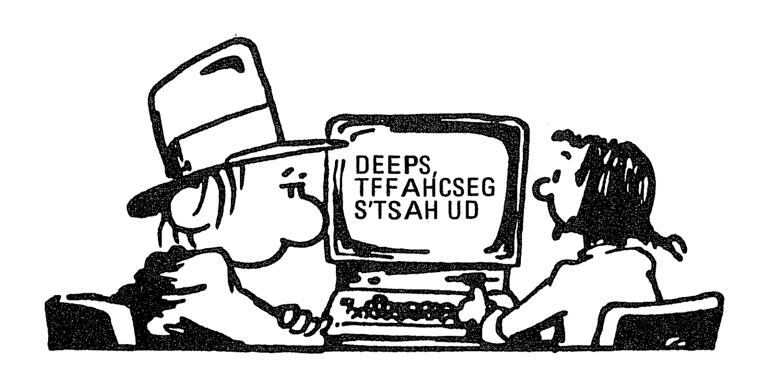
speicherten Zeichenkette nicht kennen. Die Länge kann sogar bei jeder Durchführung ohne weiteres verändert werden. Nun sind wir bereit, die Unterroutine zu schreiben:

```
2000 REM————DIE NACHRICHT UM-
     KEHREN———————
2010 FOR J = LEN(M$) TO 1 STEP —1
2020      L$ = MID$(M$,J,1)
2030      C$ = C$ + L$
2040 NEXT J
2050 RETURN
```

Die Routine hat nur wenige Schritte. Die FOR/NEXT-Schleife weist den Computer an, beginnend mit dem letzten Zeichen abwärts zu zählen, bis das erste Zeichen von M$ erreicht ist. Enthält M$ das Wort »KATZE«, wird die Schleife fünfmal durchgeführt. Beim ersten Mal entspricht J der Zahl 5, beim zweiten Mal der Zahl 4, beim dritten Mal der Zahl 3, beim vierten Mal der Zahl 2 und beim fünften Mal der Zahl 1.
L$ steht für ein Zeichen aus M$. Bleiben wir beim obengenannten Wort, so enthält L$ bei der ersten Schleifendurchführung den Buchstaben »E«, bei der zweiten »Z«, bei der dritten »T«, bei der vierten »A« und bei der fünften »K«.
Die Zeile 2030 nimmt jeweils den Wert aus L$ und fügt ihn an das Ende von C$ an. Beim Start befindet sich in C$ noch gar nichts, also ergibt hier L$ selbst die Variable C$. Am Schluß der ersten Schleife enthält C$ den Buchstaben »E«. Am Ende des zweiten Durchgangs enthält L$ den

Buchstaben »Z«, der dem Inhalt von C$ (»E«) zugeordnet wird. Danach beinhaltet C$ die Zeichenkette »EZ«. Beim dritten Durchgang wird aus L$ das »T« beigefügt, C$ besteht danach aus »EZT«. Beim vierten Durchlauf erfolgt das gleiche mit dem Buchstaben »A«, jetzt entspricht C$ dem Wort »EZTA«. Zu guter Letzt kommt das »K« hinzu, die Variable C$ enthält nun die umgedrehte Nachricht: »EZTAK«

Quiz mit dem zweiten Spieler



Nun haben wir die Originalnachricht in M$ und die verschlüsselte Nachricht in C$ gespeichert — der Rest ist einfach! Wir löschen den Bildschirm und bitten den zweiten Spieler, sich den Inhalt von C$, den wir auf dem Bildschirm ausgeben, anzusehen. Wenn er kann, soll er das richtige

Wort dann eingeben. Wir nennen die Antwort des Spielers A$. Wenn A$ dem Inhalt von M$ gleicht, dann hat der Spieler die Verschlüsselung erraten.

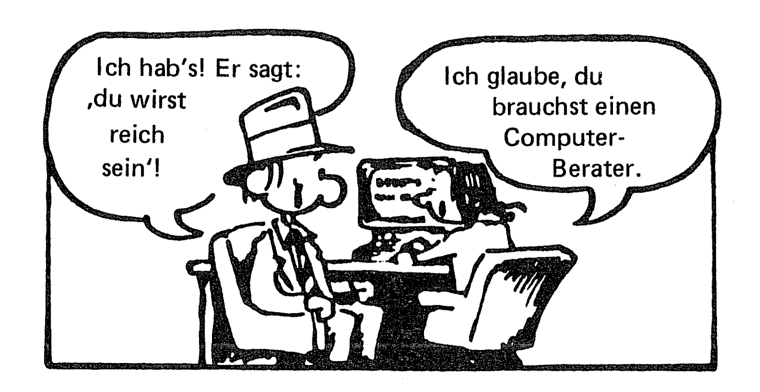
```
3000 REM----------QUIZ--------
3010 CLS    :PRINT
3020 PRINT "BITTEN SIE IHREN MITSPIE-
     LER,"
3030 PRINT "DIESE NACHRICHT ZU ENT-
     ZIFFERN"
3040 PRINT:PRINT C$
3050 PRINT:PRINT:PRINT "GEBEN SIE DIE
     ANTWORT EIN:"
3060 PRINT:INPUT A$
3070 IF A$ = M$ THEN PRINT "RICHTIG!
     DAS WAR'S!"
3080 IF A$ <> M$ THEN PRINT "SCHA-
     DE, DIE NACHRICHT WAR: ";M$
3090 RETURN
```

Das ganze Programm



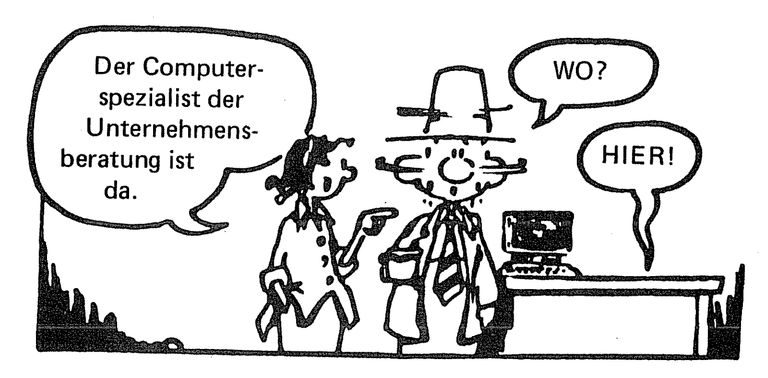

```
  10 REM————————GEHEIMSPRACHE—
     ———————
  20 REM M$   : DIE NACHRICHT
  30 REM L$   : EIN BUCHSTABE IN DER
     NACHRICHT
  40 REM C$   : DER SCHLÜSSEL
  50 REM A$   : DIE ANTWORT
  60 REM J    : EIN ZÄHLER
  70 REM————————————
 100 GOSUB 1000:REM——————DIE NACH-
     RICHT EMPFANGEN
 200 GOSUB 2000:REM———————DIE
     NACHRICHT VERSCHLÜSSELN———
 300 GOSUB 3000:REM——————QUIZ
 400 END
1000 REM——————DIE NACHRICHT EMP-
     FANGEN————————
1010 CLS   :PRINT
1020 PRINT "SAGEN  SIE IHREM MITSPIE-
     LER, ER SOLL"
```

```
1030 PRINT "WEGSEHEN, WENN SIE DIE
       NACHRICHT"
1035 PRINT "EINGEBEN"
1040 PRINT:PRINT
1050 PRINT "GEBEN SIE NUN DIE NACH-
       RICHT EIN"
1055 PRINT "DANN DRÜCKEN SIE DIE RE-
       TURN-TASTE"
1060 INPUT M$
1070 RETURN
2000 REM————DIE NACHRICHT UM-
       KEHREN————
2010 FOR J = LEN(M$) TO 1 STEP —1
2020      L$ = MID$(M$,J,1)
2030      C$ = C$ + L$
2040 NEXT J
2050 RETURN
```



```
3000 REM————————QUIZ————————
3010 CLS    :PRINT
3020 PRINT "BITTEN SIE IHREN MITSPIE-
     LER,"
3030 PRINT "DIESE NACHRICHT ZU ENT-
     ZIFFERN"
3040 PRINT:PRINT C$
3050 PRINT:PRINT:PRINT "GEBEN SIE DIE
     ANTWORT EIN:"
3060 PRINT:INPUT A$
3070 IF A$ = M$ THEN PRINT "RICHTIG!
     DAS WAR'S!"
3080 IF A$ <> M$ THEN PRINT "SCHA-
     DE, DIE NACHRICHT WAR: ";M$
3090 RETURN
```

Variationen



Dieses Spiel läßt sich sehr gut abwandeln. Zum einen könnten wir die Codierung der Nachricht verändern. Zum anderen könnten wir eine Zusammenstellung schon formulierter Nachrichten speichern, aus denen der Computer eine auswählt. Das sind zwei Dinge, die wir im nächsten Kapitel besprechen wollen. Es gibt aber auch einige kleinere Änderungen, die Sie selbständig durchführen können. Beispielsweise sind Sie in der Lage, die Ansprache durch den Computer persönlicher zu gestalten, indem Sie sich mit dem Vornamen anreden lassen. Wir wollen uns hier jedoch mit einer anderen Variation beschäftigen. Unser Spielkonzept gibt dem zweiten Mitspieler nur eine einzige Chance, die Meldung zu dechiffrieren. Meinen Sie nicht auch, es wäre besser, ihm mehr als einen Versuch zuzubilligen? Wenn ja, zeigen wir Ihnen, wie's gemacht wird. Der Beginn der QUIZ-Unterroutine (Zeilen 3000 bis 3050) bleibt bestehen: Wir geben C$ auf dem Bildschirm aus und fragen den Mitspieler nach der richtigen Antwort. Der Rest dieser Unterroutine wird jedoch dann in der Ablauflogik verändert. Anstatt mit einer Eingabe zufrieden zu sein, bauen wir eine Schleife auf, die dreimal durchgeführt wird und jedesmal eine Aussage zuläßt. Bei der Durchführung müssen wir prüfen, ob die Antwort A$ der Variablen M$ entspricht. Falls das zutrifft, das heißt, der Spieler hat die Nuß geknackt, muß der Durchlauf beendet werden.

```
3060 FOR J = 1 TO 3
3070        PRINT "VERSUCH NR: ";J
3080        INPUT A$
3090        IF A$ = M$ THEN PRINT "RICH-
     TIG!":GOTO 3130
3100        IF A$ <> M$ THEN PRINT
     "SCHADE! ";
3110 NEXT J
3120 PRINT "DIE NACHRICHT WAR: ";M$
3130 RETURN
```

Achten Sie auf das Semikolon (;) in Zeile 3100. Am Ende einer PRINT-Anweisung bewirkt dieses Zeichen, daß der nächste Text in der gleichen Zeile auf dem Bildschirm erscheint. Wenn der Spieler also bei allen Versuchen scheitert, erscheint zweimal die Nachricht:
SCHADE! VERSUCH NR: J (wobei statt J jeweils die aktuelle Zahl erscheint)
Und beim dritten Mal der Text:

SCHADE! DIE NACHRICHT WAR: M$

(hier folgt nun der Text aus M$)

3
Die Kunst der Nachrichtenverschlüsselung

Das im letzten Kapitel erarbeitete Spiel funktioniert nur, wenn ein Spieler den Computer mit einer Nachricht füttert. Jedesmal, wenn das Spiel beginnt, muß also eine neue Nachricht eingegeben werden. Das hemmt natürlich den Spielverlauf und kann auf die Teilnehmer ermüdend wirken. Ein weiteres Ärgernis wäre, daß der zweite Spieler einem über die Schulter gucken und die Nachricht vor der Verschlüsselung sehen könnte, was ihm natürlich nachher die Entzifferung erleichtern würde.

Eine bessere Idee ist wohl, eine größere Anzahl von Texten vorab im Programm zu speichern und den Computer einen dieser Texte per Zufall auswählen zu lassen. Kommt Ihnen diese Idee nicht bekannt vor?

Sollte sie eigentlich. In Kapitel 1 haben wir gelernt, wie man Texte, Nachrichten oder Zeichenketten speichert und sie später nach Zufall einzeln ausgibt, und in Kapitel 2, wie man diese Texte verschlüsselt.

Jetzt werden wir nun zwei Dinge tun: wir verbin-

den die ersten beiden Programme zu einem neuen Programm, das eine Meldung im Random-Verfahren auswählt und dann chiffriert. Anschließend vertiefen wir das Thema der Verschlüsselung durch Erfindung eines weiteren Codes oder die Benutzung zweier Codes im gleichen Programm noch etwas.
Die Steuerung, das Hauptprogramm für unser neues Programm sieht jetzt so aus:

```
10 REM—————VERSCHLÜSSELTE NACH-
   RICHTEN——————
20 REM M$(4):BEREICH FÜR VIER NACH-
   RICHTEN
30 REM J     :ZÄHLER
40 REM N     :ZUFALLSZAHL
50 REM L$    :EIN BUCHSTABE IN DER
   NACHRICHT
60 REM C$    :DIE VERSCHLÜSSELTE
   NACHRICHT
70 REM M$    :DIE ORIGINALNACHRICHT
80 REM————————
95 CLEAR 200
100 GOSUB 1000:REM—————DIE NACH-
   RICHTEN SPEICHERN
200 GOSUB 2000:REM—————EINE ZUFÄL-
   LIG AUFGREIFEN
300 GOSUB 3000:REM—————SIE VER-
   SCHLÜSSELN
400 GOSUB 4000:REM—————QUIZ
500 END
```

In Zeile 95 finden Sie ein spezielles TRS-80-Kommando, das Sie wegen der Überlänge einiger verschlüsselter Nachrichten verwenden müssen. Jedesmal, wenn Sie auf einem TRS-80 ein Programm starten, werden diesem 50 Speicherstellen für Text zugeordnet. Wenn mehr Zeichen gespeichert werden sollen, erscheint eine entsprechende Meldung (OUT OF STRING SPACE). Um das zu vermeiden, wird der CLEAR-Befehl mit Angabe der längsten möglichen Zeichenkette angewendet. In unserem Fall werden 200 Zeichen fest zugeordnet — obwohl wir vermutlich nicht die ganze Anzahl brauchen. Wenn es allerdings mehr werden, müssen Sie die Zahl im CLEAR-Befehl entsprechend erhöhen.
Eigentlich haben Sie sich schon mit jeder dieser Unterroutinen ein wenig vertraut gemacht. Die ersten beiden stammen aus dem Programm in Kapitel 1, und die letzten beiden lernten wir in Kapitel 2 kennen. Es müssen nur ganz geringfügige Änderungen vorgenommen werden.
Betrachten wir die Unterroutine 1000 in Kapitel 1, die wir »Speicherung der Vorhersagen« genannt haben: Hier speichern wir nun Nachrichten statt Weissagungen, also nennen wir den Bereich oder die Tabelle M$(4) statt F$(4). Führen Sie bitte diese Änderung durch, und schreiben Sie anstelle der Vorhersagen die von Ihnen gewünschten Nachrichten in die DATA-Zeilen. Der Rest dieser Unterroutine bleibt unverändert.
Keine Änderungen sind in der zweiten Unterroutine notwendig. Auch hier greifen wir eine Zufallszahl von 1 bis 4 auf.

Die dritte Unterroutine in unserem neuen Programm, »SIE VERSCHLÜSSELN«, das heißt, eine der Nachrichten zu chiffrieren, ist die gleiche, die wir in Kapitel 2 mit »DIE NACHRICHT UMKEHREN« bezeichnet haben. Dort war es die zweite Unterroutine, und hier ist es die dritte. Wir müssen also die Zeilennummern aus den 2000ern in die 3000er verlegen. Darüber hinaus hatten wir in Kapitel 2 nur eine Nachricht, deshalb nannten wir sie nur M$. Jetzt aber kann unsere Nachricht jede der in der Tabelle M$(4) gespeicherten Meldungen sein. Da wir gerade eine Zufallszahl N gezogen haben, ist die Nachricht, die verschlüsselt werden soll, natürlich in der Variablen M$(N) zu finden. Eine winzige Änderung sorgt für die Anpassung dieser Unterroutine. Fügen Sie bitte am Beginn der Routine die folgende Zeile ein:

```
3005 M$ = M$(N)
```

Diese Anweisung bedeutet, daß die Nachricht, die aufgegriffen wurde, im weiteren Verlauf des Programms nun M$ genannt wird. So kann der Rest der Routine unverändert bleiben.
Die letzte Unterroutine, QUIZ, ist ebenfalls identisch mit der letzten Unterroutine in Kapitel 2. Vergessen Sie aber nicht, die Zeilennumerierung von 3000 auf 4000 zu erhöhen, da wir ja in unserem neuen Programm vier Unterroutinen haben.

Eine andere Codierung



Mit einer Handvoll Änderungen haben wir zwei einfachere Programme zu einem komplizierteren zusammengefügt. Das neue Programm wird eine Nachricht per Zufallsgenerator auswählen, sie verschlüsseln und einen Mitspieler bitten, den Originaltext wiederherzustellen.
Wir können aber noch einiges mehr tun. Momentan benutzt unser Programm immer dieselbe Codierung. Wenn jemand mehrmals mit diesem Programm gespielt hat, ist er höchstwahrscheinlich in der Lage, den Code zu knakken, und der Spaß an der Sache ist vorbei. Also schreiben wir eine Unterroutine, die die Umsetzung in verschiedener Weise ermöglichen soll. Dann können wir beide Codes im Programm benutzen, wobei der Computer auch hier Zufall spielt und den Code auswählt.

Es gibt vermutlich Tausende verschiedener Codierungen, auf jeden Fall mehr, als man sich vorstellen kann. Unser erster Code kehrte einfach die Nachricht um. Ein anderer, oft benutzter Schlüssel besteht aus der Einfügung von Buchstaben zwischen die Buchstaben der Meldungen. Zum Beispiel könnte der Code für »HALLO« so aussehen: HXAXLXLXOX. Eine leichte Änderung der Unterroutine sorgt für eine solche Chiffrierung:

```
3000 REM------SIE VERSCHLÜSSELN-
     ------
3005 M$ = M$(N)
3010 FOR J = 1 TO LEN(M$)
3020      L$ = MID$(M$,J,1)
3030      C$ = C$ + L$ + "X"
3040 NEXT J
3050 RETURN
```

Zwei Änderungen müssen gemacht werden: In Zeile 3010 zählen wir nun vorwärts anstatt rückwärts, weil wir die Zeichenreihenfolge diesmal nicht umkehren wollen. In Zeile 3030 fügen wir erst einen Buchstaben der eigentlichen Nachricht und dann zusätzlich ein X bei jedem Schleifendurchlauf im Bereich C$ zusammen.
Natürlich können Sie jeden Buchstaben beziehungsweise jedes Zeichen anstelle des X verwenden. Aber ist dieser Code nicht immer noch zu leicht zu entziffern? Was halten Sie von der Idee, wenn wir mit dem Zufallsgenerator (RND) einen beliebigen Buchstaben für X in Zeile 3030

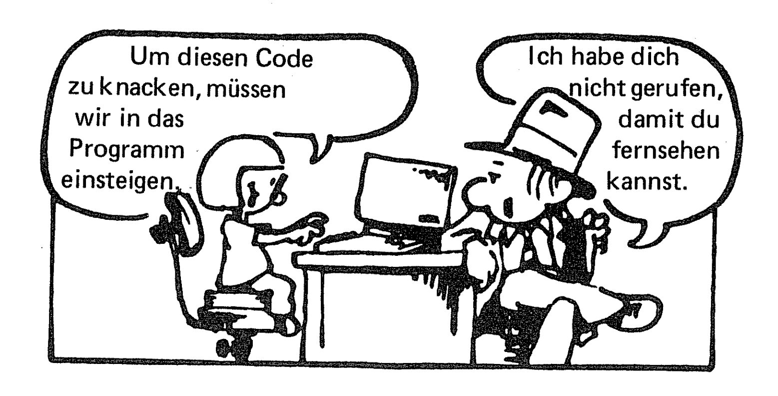
auswählten? Dann könnte HALLO nach der Codierung ungefähr so aussehen: HRAULBLIOW. Das ist dann wirklich schwer zu entziffern!
Um das zu programmieren, müssen Sie zwei weitere BASIC-Funktionen kennenlernen. Der Code wird ASCII-Code genannt und von den meisten Computern benutzt. Die ASCII-Zahl für den Buchstaben A ist 65, für B 66, für C 67 und so weiter bis Z, das die Codezahl 90 hat. Das können Sie leicht selbst überprüfen. Geben Sie beispielsweise

```
PRINT ASC ("A")
  oder
PRINT ASC ("T")
```

oder zwischen den Anführungsstrichen jeden gewünschten Buchstaben ein.
Die ASC-Funktion ändert den entsprechenden Buchstaben in die ihm zugeordnete Codezahl um. Wir aber wollen genau das Gegenteil tun — wenn wir die RND(1)-Funktion verwenden, erhalten wir eine Zufallszahl. Wir müssen diese Zahl in einen Buchstaben verwandeln. Die BASIC-Funktion CHR$ hilft uns dabei. Versuchen Sie es mit:

```
PRINT CHR$(65)
```

Der Computer wird nun den Buchstaben A wiedergeben, da A der Buchstabe mit der Nummer 65 im ASCII-Verzeichnis ist. Die CHR$-Funktion ist also das Gegenstück zur ASC-Funktion.



Nun werden wir unsere »SIE VERSCHLÜSSELN«-Unterroutine dahingehend ändern, daß wir bei jeder Schleifendurchführung eine Random-Ganzzahl von 1 bis 26 (das Alphabet hat 26 Buchstaben) erhalten. Diese Ganzzahl benutzen wir dann, um mit Hilfe der CHR$-Funktion einen Buchstaben zu bestimmen. Auf diese Weise erhalten wir »Zufallsbuchstaben«, die wir dem Code anstelle eines X jedesmal einfügen können:

```
3000 REM—————SIE VERSCHLÜSSELN—
     ——————
3005 M$ = M$(N)
3010 FOR J = 1 TO LEN(M$)
3020      L$ = MID$(M$,J,1)
3030      R = RND(26) + 64
3040      C$ = C$ + L$ + CHR$(R)
3050 NEXT J
3060 RETURN
```

Die Zeile 3030 weist den Computer an, die Zufallszahl um 64 zu erhöhen. Wir wollen ja eine Zahl zwischen 65 und 90, da dieser Zahlenbereich die Buchstaben von A bis Z im ASCII-Code darstellt. Mit irgendeiner Zahl zwischen 1 und 64 sind wir nicht zufrieden.
Außerdem werden in Zeile 3040 drei Zeichenketten zusammengefügt, und zwar jedesmal, wenn die Schleife durchgeführt wird: das, was schon codiert ist (C$), der nächste Buchstabe aus der Meldung (L$) und der zufällig ermittelte Buchstabe (CHR$(R)).

Der eine Code oder der andere?



Nun kennen wir zwei Wege der Nachrichtenverschlüsselung, und wir können die eine oder an-

dere Form für das Programm wählen. Noch besser ist es jedoch, wenn wir BEIDE im gleichen Programm benutzen können und dem Computer jedesmal die Wahl der Codierungsart überlassen. Damit das funktioniert, müssen wir beide Unterroutinen in das Programm einbauen. Da sie nicht auf derselben Zeilennummer starten können, legen wir fest, daß die Umkehrungscode-Routine auf Zeile 3000 beginnen soll und die Einfügungscode-Routine auf 3500. Und so wird's gemacht:

```
3000 REM—————UMKEHRUNGSCODE——
     —————
```

und

```
3500 REM—————EINFÜGUNGSCODE——
     ———
```

Sie kennen die Details jeder Routine bereits und können Sie also mit Leben füllen. Das gesamte Programm werden wir Ihnen später in diesem Kapitel vorstellen.
Doch zunächst muß der Computer angewiesen werden, zwischen den beiden Routinen auszuwählen. Dazu müssen wir den Steuerungsteil des Programms verändern. Das Hauptprogramm nach den REMark-Anweisungen sieht dann folgendermaßen aus:

```
100 GOSUB 1000:REM———NACHRICH-
     TENSPEICHERUNG———
```

```
200 GOSUB 2000:REM——EINE ZUFÄLLIG
    AUFGREIFEN
250 S = RND(2)
300 ON S GOSUB 3000,3500:REM——VER-
    SCHLÜSSELUNG
400 GOSUB 4000:REM————QUIZ
500 END
```

Wie Sie sehen können, benutzen wir wieder einmal die Random-Funktion. Dieses Mal, um entweder eine 1 oder eine 2 aufzugreifen und sie als »S« zu speichern. Wenn S den Wert 1 annimmt, soll das Programm die Unterroutine 3000 durchführen, wenn S eine 2 beinhaltet, sollte die Unterroutine 3500 abgearbeitet werden.
Die ON————GOSUB-ANWEISUNG in Zeile 300 bewirkt genau das. Sie bedeutet bei S gleich 1 die Verzweigung auf die erste angeführte Unterroutine und bei S gleich 2 die zur zweiten. Falls es sich um drei verschiedene Codierungs-Unterroutinen handeln würde und die letzte würde bei Zeile 3800 beginnen, könnten wir für S die Zufallszahlen 1, 2 und 3 bestimmen und die Zeile 300 wie folgt ändern:

```
300 ON S GOSUB 3000,3500,3800
```

Das ganze Programm



Der folgende Abschnitt zeigt die letzte Version unseres Programms. Es speichert ein paar Meldungen in einer Tabelle, greift eine von ihnen per Zufallsgenerator für die Codierung heraus, codiert sie unter Benutzung einer der zwei Methoden und bittet den Mitspieler um einen Dechiffrierungsversuch.

```
10 REM————VERSCHLÜSSELTE NACH-
   RICHTEN——————
20 REM M$(4) :BEREICH FÜR VIER
   NACHRICHTEN
30 REM J       :ZÄHLER
40 REM N       :ZUFALLSZAHL NACH-
   RICHTEN
50 REM S       :ZUFALLSZAHL UNTER-
   ROUTINE
```

```
  60 REM R     :ZUFALLSZAHL BUCH-
     STABE
  70 REM L$    :EIN BUCHSTABE IN DER
     NACHRICHT
  80 REM C$    :DIE VERSCHLÜSSELTE
     NACHRICHT
  85 REM M$    :DIE ORIGINALNACHRICHT
  90 REM———————
  95 CLEAR 200
 100 GOSUB 1000:REM———————NACH-
     RICHTEN SPEICHERN—
 200 GOSUB 2000:REM———EINE ZUFÄL-
     LIG AUFGREIFEN
 250 S = RND(2)
 300 ON S GOSUB 3000,3500:REM——
     VERSCHLÜSSELUNG
 400 GOSUB 4000:REM—————QUIZ——
     ——
 500 END
1000 REM—————NACHRICHTEN SPEI-
     CHERN——————
1010 DATA "Einen schönen Tag wünsche
     ich"
1020 DATA "Wir sehen uns morgen"
1030 DATA "Wir treffen uns an der alten
     Eiche"
1040 DATA "Dies ist eine geheime Nach-
     richt"
1050 DIM M$(4)
1060 FOR J = 1 TO 4
1070      READ M$(J)
1080 NEXT J
1090 RETURN
```

```
2000 REM—————NACHRICHTENZUFALLS-
     ZAHL————
2010 N = RND(4)
2020 RETURN
3000 REM—————UMKEHRUNGSCODE——
     ———
3005 M$ = M$(N)
3010 FOR J = LEN(M$) TO 1 STEP —1
3020      L$ = MID$(M$,J,1)
3030      C$ = C$ +L$
3040 NEXT J
3050 RETURN
3500 REM—————EINFÜGUNGSCODE———
     ——
3505 M$ = M$(N)
3510 FOR J = 1 TO LEN(M$)
3520      L$ = MID$(M$,J,1)
3530      R = RND(26) + 64
3540      C$ = C$ + L$ + CHR$(R)
3550 NEXT J
3560 RETURN
4000 REM———————QUIZ———————
4010 CLS
4020 PRINT "BITTEN SIE IHREN MITSPIE-
     LER,"
4030 PRINT "DIESE NACHRICHT ZU ENT-
     ZIFFERN"
4040 PRINT:PRINT:PRINT C$
4050 PRINT:PRINT "GEBEN SIE DIE ANT-
     WORT EIN:"
4060 INPUT A$
4070 IF A$ = M$ THEN PRINT "RICHTIG!
     DAS WAR'S!"
```

```
4080 IF A$ <> M$ THEN PRINT "SCHA-
     DE, DIE NACHRICHT WAR: ";M$
4090 RETURN
```

Dieses Programm ist schon ganz schön lang, aber da wir es Abschnitt für Abschnitt zusammengebaut haben, dürfte es für Sie nicht schwierig gewesen sein, den einzelnen Schritten zu folgen. Beachten Sie bitte, daß wir die Random-Funktion RND jetzt insgesamt dreimal nutzen: Wir wählen die Nachricht durch Random aus, wir ermitteln die Codierungsunterroutine durch Random, und wir fügen die mit Random selektierten Buchstaben in die Nachricht ein.

Variationen



Nachdem Sie alle für dieses Programm notwendigen Schritte kennen, möchten wir Ihnen einige Erweiterungen vorschlagen, mit deren Hilfe Sie selbständig das Programm ausbauen können:

1. Verändern Sie die Anzahl der in der ersten Unterroutine gespeicherten Nachrichten, und lassen Sie sie vom Programm auswählen.

2. Wählen Sie mehr als eine Nachricht in einem Durchlauf zur Entzifferung aus.

3. Geben Sie dem Spieler mehr als eine Chance, die Nachricht(en) zu entschlüsseln.

Zwei weitere Varianten, die wir bisher noch nicht erwähnt haben, die aber leicht in das Programm einzubringen sind, wollen wir Ihnen nicht vorenthalten:

1. Anstatt sich darauf zu beschränken, Buchstaben zwischen die Zeichen der Nachricht ein-

zufügen, können Sie jedes Tastatursymbol im Zufallsgenerator zulassen — eingeschlossen den Stern und die Anführungsstriche. HINWEIS: Überprüfen Sie die bei CHR$(33) startenden Zeichen bis zur CHR$(65). Das ist wichtig, weil wir in Zeile 3030 die Zahl 64 zur Zufallszahl addiert haben, da die Codezahl 65 in ASCII auf den Buchstaben A verweist.

2. Wenn Sie das Programm durchführen, werden Sie feststellen, daß ab dem zweiten Wort einer Nachricht schon vor dem ersten Buchstaben ein Zufallszeichen generiert wird. Dieses Phänomen ist leicht erklärbar. Eine Leerstelle zwischen zwei Worten wird vom Computer ebenso als Zeichen betrachtet wir ein normales sichtbares Zeichen — unsere Unterroutine wirft also hinter jeder Leerstelle auch ein Zufallszeichen aus. Wenn Sie diesen Nebeneffekt nicht mögen, können Sie den Computer anweisen, Leerstellen zu ignorieren, sobald sie auftreten. Sie brauchen nur die EINFÜGUNGSCODE-Unterroutine um folgendes Kommando zu ergänzen:

```
3525 IF L$ = CHR$(32) THEN C$ = C$ +
     L$:GOTO 3550
```

CHR$(32) ist die ASCII-Codezahl für eine Leerstelle. Die oben angegebene Zeile weist den Computer an, falls in der Originalnachricht eine Leerstelle auftritt, nur diese der verschlüsselten Nachricht beizufügen. Das Zufallszeichen wird in diesem Fall ausgelassen und die Arbeit mit dem nächsten Zeichen fortgesetzt.

4
Das Rennen der Anfangsbuchstaben

Sozusagen »spielend« haben wir einige wichtige Techniken gelernt: verschiedene Arten, eine Codierung vorzunehmen, Nachrichten zu speichern und auszugeben und die Random-Funktion zu nutzen. Diese Funktion werden Sie immer und immer wieder in den verschiedensten Spielen finden. Aber bis jetzt haben wir eine Spezialität des Computers völlig außer acht gelassen: die Animation. Bei dieser Spieltechnik geht es um die Bewegung von Bildern auf dem Bildschirm.
Viele Computerspiele sind ohne Animation nicht denkbar. Programmierer verwenden Jahre dar-



auf, die fortschrittlichsten neuen Animationstechniken zu studieren. In diesem Buch wollen wir Sie in die Animation von Zeichen einführen — das heißt, die Benutzung eines beliebigen Symbols der Tastatur und dessen scheinbare Bewegung auf dem Bildschirm.
Der TRS-80 besitzt mehrere Möglichkeiten der Animation. In diesem Kapitel lernen wir die spezielle Graphikdarstellung des TRS-80 kennen: Sie können den Computer anweisen, auf jeder gewünschten Bildschirmposition kleine Rechtecke darzustellen. Wenn man nun ein solches Rechteck an einer bestimmten Stelle aufleuchten läßt, anschließend wieder löscht und unmittelbar daneben wieder anzeigt, dann scheint sich dieses Rechteck zu bewegen.
Wir beginnen mit einem Rennen zwischen zwei Teilnehmern. Auf der linken Seite des Schirms plazieren wir die Anfangsbuchstaben von zwei Spielern, auch Initiale genannt, daneben erscheint jeweils ein Lichtblock in seiner Ausgangsposition. Von dort bewegen sich die Rechtecke mit unterschiedlicher Geschwindigkeit nach rechts, sobald einer der Spieler die RETURN-Taste als Startzeichen gedrückt hat. Wenn eines der beiden Zeichen die Ziellinie erreicht, zeigt das Programm an, wer gewonnen hat. Jedesmal, wenn Sie dieses Programm starten, wird per Random eine Geschwindigkeit für jeden Spieler ermittelt, so daß jeder zu jeder Zeit gewinnen kann.
Im nachstehenden Hauptprogramm sind die vier wichtigsten Teile (Unterroutinen) ebenso

wie die Variablen, die wir benutzen werden, aufgeführt:

```
 10 REM-----ANFANGSBUCHSTABEN-
    RENNEN-----
 20 REM A$   : DER ERSTE ANFANGS-
    BUCHSTABE
 30 REM B$   : DER ZWEITE ANFANGS-
    BUCHSTABE
 40 REM SA   : GESCHWINDIGKEIT VON
    RECHTECK A$
 50 REM SB   : GESCHWINDIGKEIT VON
    RECHTECK B$
 60 REM PA   : POSITION VON RECHTECK
    A$
 70 REM PB   : POSITION VON RECHTECK
    B$
 75 REM W$   : DER GEWINNER
 80 REM R$   : EINE ANTWORT (START
    DES RENNENS)
 85 REM X    : EIN ZÄHLER
 90 REM-----------
100 GOSUB 1000:REM---EINGABE AN-
    FANGSBUCHSTABEN
200 GOSUB 2000:REM---AUFBAU DES
    BILDSCHIRMS
300 GOSUB 3000:REM---DIE GE-
    SCHWINDIGKEIT
400 GOSUB 4000:REM---DAS RENNEN
500 END
```

Eingabe der Initialen



Zuallererst benötigen wir zwei Anfangsbuchstaben, die das Rennen gegeneinander austragen. Alles andere in der folgenden Routine sollte Ihnen schon bekannt sein:

```
1000 REM———EINGABE DER ANFANGS-
     BUCHSTABEN——————
1010 CLS:PRINT "GEBEN SIE DEN AN-
     FANGSBUCHSTABEN DES ERSTEN
     SPIELERS EIN:"
1020 INPUT A$
1030 PRINT:PRINT "NUN GEBEN SIE DEN
     ANFANGSBUCHSTABEN DES ZWEI-
     TEN SPIELERS:"
1040 INPUT B$
1050 RETURN
```

Wir löschen den Bildschirm und fragen nach den jeweiligen Anfangsbuchstaben der beiden Spieler, die wir in der nächsten Unterroutine verwenden wollen.

Aufbau des Bildschirms



In diesem Abschnitt müssen wir uns zuerst einmal mit der Frage befassen, wie ein TRS-80-Bildschirm gestaltet wird. Während Sie dieses Kapitel lesen, sollten Sie zusätzlich das Bildschirmdiagramm in Ihrem TRS-80-Handbuch beachten. Der TRS-80 verfügt über 16 Zeilen vom oberen bis zum unteren Rand des Bildschirms. In dieser Unterroutine geben wir den ersten Anfangsbuchstaben am Anfang der dritten Zeile von oben und den zweiten auf der gleichen Stelle in Zeile 5 aus. Die Zeile 2010 löscht den Bildschirm, über-

springt zwei Zeilen und zeigt A$ in der dritten Zeile an. Zeile 2020 setzt zwei weitere Zeilen vor und läßt B$ in der fünften von oben erscheinen. Wichtig zu wissen ist auch, daß der Bildschirm in 6.144 kleine Rechtecke eingeteilt ist, die Sie an jeder beliebigen Stelle aufleuchten und verschwinden lassen können. Diese Bildschirmpositionen sind matrixförmig in der Horizontalen von 0 bis 127 und in der Vertikalen von 0 bis 47 durchnummeriert. Das Kommando

SET (0,0)

gibt auf der äußersten Stelle oben links ein Rechteck aus.

SET (127,47)

zeigt einen solchen Lichtblock auf der untersten Position ganz rechts.
In Zeile 2030 lassen wir das Zeichen aufleuchten (10,7), das heißt, zehn Einheiten vom linken und sieben Einheiten vom oberen Bildschirmrand entfernt. Damit steht es genau neben dem ersten Anfangsbuchstaben. Anschließend zeigt der Computer ein Rechteck auf (10,13), also wieder zehn Einheiten vom linken, aber diesmal dreizehn vom oberen Rand entfernt. Diese Position befindet sich neben dem zweiten Initial.
Probieren Sie in jedem Fall auch mal andere Bildschirmstellen aus. Benutzen Sie dabei aber bitte eines der TRS-80-Formulare für die Bildschirmausgabe, mit dem Sie die Anzeige des Rechtecks vorher genau planen können.

```
2000 REM———AUFBAU DES BILD-
     SCHIRMS———————
2010 CLS:PRINT:PRINT:PRINT A$
2020 PRINT:PRINT B$
2030 SET(10,7):SET(10,13)
2040 PRINT:PRINT
2050 PRINT "DRÜCKEN SIE DIE RETURN-
     TASTE ZUM START DES RENNENS:"
2060 INPUT R$
2070 RETURN
```

Animation — die Kunst der Bewegung



Bevor wir uns mit dem Rennen selbst beschäftigen, sollten wir untersuchen, was Animation für einen Computer bedeutet. Wie schon gesagt, gibt es viele Arten der Animation. Wir spezialisie-

ren uns hier auf die Bewegung eines Zeichens quer über den Bildschirm mittels der vorhandenen TRS-80-Graphikfunktion.
Wie wir es schon früher bei neuer Ablauflogik getan haben, werden wir nun ein kleines Testprogramm schreiben. Wenn Sie gerade das Programm »Rennen« eingeben, sichern Sie es bitte und geben NEW ein, bevor Sie dieses Testprogramm erfassen.
Animation besteht grundsätzlich aus der Wiederholung mehrerer Schritte: Wir bringen Menschen dazu, zu glauben, daß sich etwas über den Bildschirm bewegt, indem wir ein Zeichen an einer bestimmten Stelle des Bildschirms ausgeben, es wieder löschen und an einer anderen Stelle, ein wenig entfernt, wieder ausgeben. Durch die ständige Wiederholung dieser Schritte in kurzen Abständen scheint sich das Zeichen zu bewegen — zu »laufen«. Und mit dieser Schrittfolge führt man die Animation durch:

1. Ausgabe eines Zeichens auf dem Bildschirm.
2. Wiederholung der folgenden Schritte in einer bestimmten Anzahl:
a) Löschen des Zeichens.
b) Erneute Ausgabe des Zeichens in einer geringen Entfernung.
Versuchen Sie es einmal mit dem folgenden Animations-Testprogramm:

```
10 CLS
20 SET (45,20)
30 FOR X = 45 TO 95
```

```
40      RESET (X,20)
50      SET (X+1,20)
60      FOR J = 1 TO 100:NEXT J
70 NEXT X
80 END
```

Zeile 10 löscht den Bildschirm. Die Zeile 20 bedeutet: Gehe dahin, wo das kleine Rechteck erscheinen soll, und zeige es dort, das heißt 45 von links und 20 von oben. Die Schleife in den Zeilen 30 bis 70 erhöht die Variable X von 45 auf 95. Bei jedem Schleifendurchlauf wird mit RESET in Zeile 40 das Rechteck gelöscht und mit SET in Zeile 50 eine Bildschirmstelle weiter rechts wieder ausgegeben. Bei der letzten Durchführung enthält X den Wert 95, also wird das Zeichen auf 95,20 gelöscht und auf 96,20 (96 entspricht X+1) wieder sichtbar.
Die Zeile 60 stellt eine »Pause« für den Computer dar. Dieser Stillstand wird durch einen einfachen Zählvorgang von 1 bis 100 erreicht. Ohne diesen Befehl würde der Lichtblock viel zu schnell den Bildschirm überqueren. Während des »Leerlaufs« zählt der Computer, ohne etwas anderes zu tun. Diese Tatsache bewirkt eine Verlangsamung der Bewegung. Verändern Sie doch einmal die Zahl 100 in Zeile 60 und beobachten den Effekt!
Unsere gesamte Animation enthält die gleiche Art von Schritten. Allerdings müssen wir jeweils Anpassungen vornehmen, die sich aus der Situation ergeben. Manchmal wissen wir nicht im voraus, wie weit wir das Zeichen bewegen wollen.

Manchmal, wie beispielsweise später in diesem Kapitel, haben wir zwei Zeichen, die wir bewegen wollen, und zwar jeweils um eine unterschiedliche Anzahl von Spalten bei jedem Bewegungsschritt. In welcher der beiden Situationen Sie sich auch immer befinden, denken Sie daran, daß wir ständig die gleichen Schritte wiederholen: Wir löschen das Zeichen, geben es ein Stück weiter wieder aus und machen eine Pause. Die einzige tatsächliche Schwierigkeit, auf die wir in diesem Rennen treffen, besteht darin, das »Stück weiter« herauszufinden. Und hierfür benutzen wir wieder eine Zufallszahl.

Die Wahl der Geschwindigkeit



Animation entsteht also durch die Ausgabe eines Zeichens an einem bestimmten Punkt des

Bildschirms, seine Löschung, die erneute Ausgabe in einer kurzen Entfernung, seine Löschung, die erneute Ausgabe in einer kurzen Entfernung...
Auf dem Bildschirm sehen wir die beiden Rechtecke in unterschiedlichen Zeilen in der Spalte 10. Wenn wir den Lichtblock A löschen und ihn in der Spalte 12 wieder ausgeben, hat er sich zwei Stellen weit bewegt. Wenn wir das Rechteck B in Spalte 11 ausgeben, dann ist es nur um eine Stelle vorgerückt.
Je mehr Stellen zwischen der ursprünglichen Position und der neuen liegen, um so schneller scheint sich das Zeichen zu bewegen. Deshalb werden wir Zufallszahlen zu Hilfe nehmen, um die Bewegungsgeschwindigkeit von A und B zu bestimmen. Die Zahlen können entweder 1, 2 oder 3 sein, da wir den Rechtecken eine Geschwindigkeit von 1, 2 oder 3 geben wollen.

```
3000 REM———AUSWAHL DER GE-
     SCHWINDIGKEIT————
3010 SA = RND(3)
3020 SB = RND(3)
3030 RETURN
```

Bei jedem Programmlauf können SA und SB 1, 2 oder 3 sein. In der Unterroutine DAS RENNEN addieren wir jeweils SA und SB zur derzeitigen Position des betreffenden Rechtecks. Damit erfahren wir die Stelle, auf der das Zeichen im nächsten Schritt erscheinen soll. Wenn SA den Wert 3 enthält und die zur Zeit eingenommene Position

des Lichtblocks ist 10, dann wird er in dieser Spalte gelöscht und anschließend in der dreizehnten angezeigt, im folgenden in sechzehn und so weiter. Beinhaltet SB die Zahl 1, wird das zweite Symbol zu einem Zeitpunkt immer nur um eine Einheit vorrücken, und das Initial A wird das Rennen gewinnen.

Das Rennen



Die Speicherung der Geschwindigkeit von A und B genügt nicht. Wir müssen ebenfalls jede erreichte Bildschirmposition der beiden Rechtecke speichern. Wenn wir das nicht tun, wie sollen wir dann erkennen können, welches von beiden gewonnen hat? Nehmen wir an, wir haben beschlossen, daß derjenige Anfangsbuchstabe gewinnt, dessen Symbol als erstes die Position 90 überschreitet. Wir kennen die Startpositionen

(Spalte 10) von A und B, aber da die Geschwindigkeit der beiden durch ein Randomverfahren ermittelt wird, wissen wir nicht, welcher Lichtblock sich um zwei oder nur um eine, zwei oder drei Stellen weiterbewegt hat. Deshalb benötigen wir zwei Variable, PA und PB, die für die Position von A und B stehen. Diese geben uns Auskunft über die durch die jeweils letzte Bewegung erreichte Position.

Und noch etwas: Es wäre eine nette Geste, wenn nach Beendigung des Rennens der Gewinner durch das Programm genannt würde. Dafür belegen wir die Variable W$, in der wir den Anfangsbuchstaben speichern, der als erster die Ziellinie überschreitet.

Zur besseren Übersicht teilen wir diese Unterroutine in kleine Bausteine auf und untersuchen jeden Abschnitt einzeln:

```
4000 REM———DAS RENNEN——————
4010 PA = 10:PB = 10
```

Wir beginnen mit der Initialisierung, also der erstmaligen Festsetzung der Position jedes Rechtecks durch den Wert 10, da wir sie bereits in der Spalte 10 ausgegeben haben. Anschließend bewegen wir das Rechteck A:

```
4020 RESET (PA,7)
4030 PA = PA + SA
4040 SET (PA,7)
4050 IF PA > 90 THEN W$ = A$:GOTO
     4120
```

Die ersten drei Zeilen sorgen für die Bewegung, die letzte Zeile dient zur Überprüfung, ob der Anfangsbuchstabe das Rennen gewonnen hat.
Die Zeile 4020 positioniert mit RESET auf Zeile 7 und auf die aktuelle Spalte, in der sich das Rechteck zur Zeit befindet, und löscht das Zeichen auf dem Bildschirm. Die Zeile 4030 addiert in PA die Anzahl Stellen, die A weitergehen soll. PA enthält am Anfang 10. Wenn SA eine 2 enthält (abhängig von der RND-Funktion), dann hat PA den Wert 10 + 2, also 12. Die Befehlszeile 4040 mit dem SET weist den Computer an, in welcher Spalte er das Symbol jetzt ausgeben soll.
Wir benutzen die gleichen Befehle immer und immer wieder, um den Anfangsbuchstaben A$ zu bewegen: Er wird auf seiner letzten Position gelöscht, die neue Position wird berechnet, und er wird an dieser Stelle ausgegeben.
Die Zeile 4050 prüft, ob PA einen größeren Wert als 90 enthält. Ist das der Fall, dann ist A$ der Gewinner, und wir können in den Teil des Programms übergehen, in dem der Gewinner angekündigt wird. Ist PA noch nicht größer als 90, werden die von dieser Bedingung abhängigen Befehle einfach ignoriert, und wir fahren mit der Bewegung von B fort:

```
4060 RESET (PB,13)
4070 PB = PB + SB
4080 SET (PB,13)
4090 IF PB > 90 THEN W$ = B$:GOTO
     4120
```

Diese vier Zeilen sind identisch mit den Zeilen, die der Fortbewegung von A dienen, mit Ausnahme der Tatsache, daß wir hier die für B bestimmten Variablen benutzen und die RESET- und die SET-Anweisung sich auf die Zeile beziehen, in der sich B bewegt. Wir löschen das Rechteck B in seiner ursprünglichen Position, ermitteln die neue Stelle und geben es dort aus. Anschließend prüfen wir, ob es bereits das Rennen gewonnen hat. Wenn nicht, fahren wir fort:

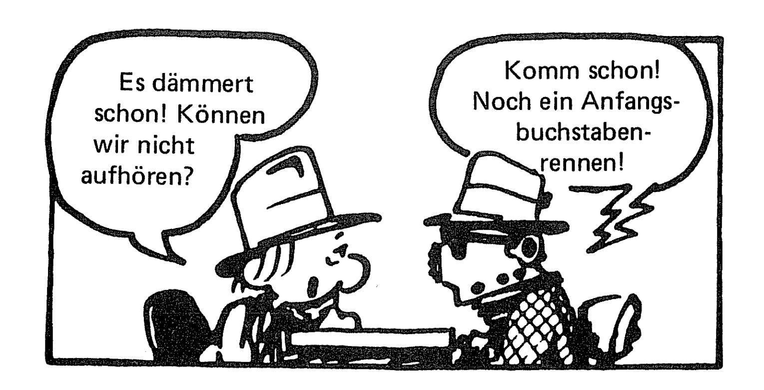
```
4100 FOR X = 1 TO 40:NEXT X
4110 GOTO 4020
```

Die Zeile 4100 besteht aus einer »Pausen«-Schleife, die vierzigmal durchlaufen wird. In dieser Zeit pausiert das Programm. Da Computer bekanntlich sehr schnell arbeiten, würde man ohne diese Pause das Rennen überhaupt nicht verfolgen können, da die Symbole nur so über den Schirm zischen würden. Sie können gerne mit der Länge der Pause experimentieren, und zwar durch Veränderung der Zahl 40 in einen von Ihnen beliebig bestimmten Wert.
Die Zeile 4110 verweist das Programm zurück auf den Teil, in dem das Zeichen A bewegt wird. Hierdurch entsteht eine Schleife. Wir bewegen das Rechteck A und prüfen, ob es gewonnen hat, bewegen dann das Rechteck B und prüfen, und so weiter, bis entweder das eine oder das andere die letzte Spalte (90) überschritten hat. Dann verzweigt das Programm zur Zeile 4120:

```
4120 PRINT:PRINT "DER GEWINNER IST:
     ";W$
4130 RETURN
```

und das Rennen ist beendet.

Das gesamte Programm




```
10 REM-----ANFANGSBUCHSTABEN-
   RENNEN------
20 REM A$   : DER ERSTE ANFANGS-
   BUCHSTABE
30 REM B$   : DER ZWEITE ANFANGS-
   BUCHSTABE
40 REM SA   : GESCHWINDIGKEIT VON
   RECHTECK A$
50 REM SB   : GESCHWINDIGKEIT VON
   RECHTECK B$
```

```
60 REM PA     : POSITION VON RECHTECK
   A$
70 REM PB     : POSITION VON RECHTECK
   B$
75 REM W$     : DER GEWINNER
80 REM R$     : EINE ANTWORT (START
   DES RENNENS)
85 REM X      : EIN ZÄHLER
90 REM——————
100 GOSUB 1000:REM———EINGABE AN-
    FANGSBUCHSTABEN
200 GOSUB 2000:REM———AUFBAU DES
    BILDSCHIRMS
300 GOSUB 3000:REM———DIE GE-
    SCHWINDIGKEIT
400 GOSUB 4000:REM———DAS RENNEN
500 END
1000 REM———EINGABE DER ANFANGS-
     BUCHSTABEN——————
1010 CLS:PRINT "GEBEN SIE DEN AN-
     FANGSBUCHSTABEN DES ERSTEN
     SPIELERS EIN:"
1020 INPUT A$
1030 PRINT:PRINT:PRINT "NUN DEN AN-
     FANGSBUCHSTABEN DES ZWEITEN
     SPIELERS:"
1040 INPUT B$
1050 RETURN
2000 REM———AUFBAU DES BILD-
     SCHIRMS—————
2010 CLS:PRINT:PRINT:PRINT A$
2020 PRINT:PRINT B$
2030 SET(10,7):SET(10,13)
```

```
2040 PRINT:PRINT
2050 PRINT "DRÜCKEN SIE DIE RETURN-
     TASTE ZUM START DES RENNENS:"
2060 INPUT R$
2070 RETURN
3000 REM———AUSWAHL DER GE-
     SCHWINDIGKEIT——————
3010 SA = RND(3)
3020 SB = RND(3)
3030 RETURN
4000 REM———DAS RENNEN——————
     ——
4010 PA = 10:PB = 10
4020 RESET (PA,7)
4030 PA = PA + SA
4040 SET (PA,7)
4050 IF PA > 90 THEN W$ = A$:GOTO
     4120
4060 RESET (PB,13)
4070 PB = PB + SB
4080 SET (PB,13)
4090 IF PB > 90 THEN W$ = B$:GOTO
     4120
4100 FOR X = 1 TO 40:NEXT X
4100 GOTO 4020
4120 PRINT:PRINT "DER GEWINNER IST:
     ";W$
4130 RETURN
```

Variationen



Wenn Sie einmal wissen, wie der Hase läuft, fällt Ihnen die Animation immer leichter und leichter. Alles, was Sie brauchen, ist Praxis. Und damit Sie sich nicht langweilen, machen wir Ihnen noch vier Vorschläge zur Optimierung des Programms. Sie haben sicher Phantasie genug, um sich noch weitere auszudenken.

1. Bildschirmformat: Gestalten Sie den Schirm nach Ihren Vorstellungen durch Ausgabe eines Spieltitels oben auf dem Bildschirm, durch Markierung der Ziellinie, durch Zeichnung eines Rahmens rund um die Rennstrecke. Sie sollten dabei Ihrer Phantasie freien Lauf lassen.

2. Höhere Geschwindigkeit: Wir haben nur eine Geschwindigkeit von 1, 2 oder 3 gewählt. Sie wünschen sich vielleicht weitere Variationen. Wissen Sie noch, wie Sie die Random-Funktion

verwenden müssen, um den Bereich der verfügbaren Zahlen zu verändern? Vor allen Dingen denken Sie daran: Wenn Sie die Geschwindigkeit zu hoch ansetzen, sieht es aus, als ob der Anfangsbuchstabe über den Bildschirm springt. Sie können aber auch die Ziellinie verändern oder die Pause in Zeile 4100.

3. Mehr Teilnehmer am Rennen: Was halten Sie von einem dritten Anfangsbuchstaben als weiteren Rennteilnehmer? Sie müssen bloß drei eingeben, drei Geschwindigkeiten festsetzen, sie auf ihre Startpostitionen ausgeben und alle drei zur selben Zeit bewegen. Benutzen Sie die gleichen vier Schritte in der RENNEN-Unterroutine für das dritte Rechteck, wie wir sie für die beiden anderen verwendet haben. Sie können sogar einen vierten oder einen fünften Teilnehmer mitspielen lassen.

4. Rennen rückwärts: Sie können das Rennen auch von der rechten zur linken Seite des Bildschirms laufen lassen. Hierzu müssen Sie einige Zahlen verändern. Beim BILDSCHIRMAUFBAU positionieren Sie mit SET auf eine hohe Spaltennummer, sagen wir zum Beispiel 100, und geben Sie die Symbole dort aus. Deshalb müssen Sie auch in der RENNEN-Routine die Variablen PA und PB mit 100 initialisieren. Anschließend wird rückwärts gezählt:

```
4030 PA = PA — SA
4070 PB = PB — SB
```

Vergessen Sie nicht zu überprüfen, ob die Posi-

tion des jeweiligen Anfangsbuchstabens KLEINER ist als die Ziellinie:

```
4050 IF PA < 10 THEN W$ = A$:GOTO
     4120
4090 IF PB < 10 THEN W$ = B$:GOTO
     4120
```

Und jetzt eine Herausforderung: Können Sie das Rennen von der obersten Zeile des Bildschirms zur untersten durchführen? Dazu müssen Sie bestimmt nicht viel ändern. Aber beachten Sie bitte, daß statt der Spaltennummern, die sich in diesem Fall nicht verändern, die Zeilennummern bei jeder Bewegung neue Werte erhalten müssen, da die Anfangsbuchstaben sich ja von Zeile zu Zeile, also senkrecht, bewegen und nicht, wie in unserem jetzigen Programm, von Spalte zu Spalte.

5
Die Seifenblase platzt

Das Rennen der Anfangsbuchstaben war noch ein vergleichsweise einfaches Spiel — es bestand eigentlich nur aus dem Rennen. Animation kann aber ebensogut als Teil komplizierterer Spiele auftreten. Stellen Sie sich folgenden Bildschirminhalt vor: Auf der rechten Seite befindet sich ein »Ballon« oder eine »Seifenblase« und auf der linken der Anfangsbuchstabe eines Spielers. In der unteren Hälfte des Bildschirms wird dem Spieler ein arithmetisches Problem genannt, das er oder sie lösen muß. Wenn die Antwort richtig ist, bewegt sich der Anfangsbuchstabe über den Bildschirm und bringt die »Blase«, in der sich eine Nachricht befindet, zum Platzen.

Dieses Spiel ist eine Kombination aus vielen Elementen, die wir bereits von früheren Programmen her kennen: Animation, Zufallszahlen, die Benutzung von DATA-Anweisungen, die Möglichkeit mehrerer Antwortversuche. Das folgende Programm weicht von dem im letzten Kapitel erarbeiteten insofern ab, daß wir nicht ein Lichtsymbol über den Bildschirm bewegen, sondern den Anfangsbuchstaben des Spielers selbst: Die Seifenblase wird durch die leuchtenden Recht-

ecke dargestellt, wir werden jedoch durch Darstellung und Löschung einen Buchstaben in Bewegung setzen. Aber keiner dieser Bausteine muß in sich selbst kompliziert sein.

```
10 REM————DIE SEIFENBLASE PLATZT—
   —————
20 REM I$    : ANFANGSBUCHSTABE
30 REM H     : HORIZONTALE POSITION
40 REM V     : VERTIKALE POSITION
50 REM X     : ZUFALLSZAHL FÜR DAS
   PROBLEM
60 REM Y     : ZUFALLSZAHL FÜR DAS
   PROBLEM
70 REM A     : ANTWORT
80 REM J     : ZÄHLER
90 REM——————————
100 GOSUB 1000:REM——BILDSCHIRM-
    AUFBAU———————
200 GOSUB 2000:REM——PROBLEM (S)—
    ———————
300 GOSUB 3000:REM——PLATZEN DER
    SEIFENBLASE——————
400 END
```

Der Bildschirmaufbau



Diese Unterroutine enthält mehrere kleine Teilbereiche. Zuerst geben wir diesem Spiel einen Namen und setzen den Titel an den oberen Rand des Bildschirms. Dann zeichnen wir den Ballon und geben den Anfangsbuchstaben ein. Zum Schluß erklären wir dem Mitstreiter das Spiel. Zum besseren Verständnis sehen wir uns jeden Teilbereich im einzelnen an:

```
1000 REM—————BILDSCHIRMAUFBAU—
     ——————
1010 CLS
1020 PRINT @20,"DIE  SEIFENBLASE
     PLATZT"
```

Das ist eine neue Form des PRINT-Befehls. Bei der Benutzung des PRINT @ (das heißt PRINT AT) können Sie Zeichen auf jeder Bildschirmposition

ausgeben. Wir wollen uns nun etwas eingehender mit diesem Thema befassen, deshalb sollten Sie ein Arbeitsformular für Bildschirmausgabe am TRS-80 zu Rate ziehen, falls Sie eins zu Hand haben.
Der Bildschirm ist eingeteilt in sechzehn Zeilen mit jeweils 64 Spalten, also stehen insgesamt 16 x 64 Positionen zur Verfügung, auf denen ein Zeichen darstellbar ist. Wenn wir links oben beginnen, heißt die erste Adresse 0, die nächste 1 und so weiter bis zur letzten rechts unten. Die erste Zeile besteht aus den Stellen 0 bis 63, die zweite aus 64 bis 127, die dritte aus 128 bis 191. Die allerletzte Position trägt die Nummer 1024.
Mit dieser Information können Sie Ihrem Computer genau mitteilen, auf welcher Bildschirmstelle Sie die Ausgabe einer Nachricht oder eines Zeichens wünschen. Versuchen Sie es doch einmal mit dem folgenden Beispiel:

```
PRINT @475 ,"HALLO"
```

Das angegebenen Wort wird ziemlich genau in der Mitte des Bildschirms, nämlich in der achten Zeile, erscheinen. Das oben dargestellte Kommando (Zeile 1020) stellt den Text »DIE SEIFENBLASE PLATZT« ab Ausgabeposition 20 dar, das heißt, zwanzig Stellen von der ersten links oben entfernt.
Im nächsten Schritt zeichnen wir die Seifenblase durch die Anzeige von 28 rechteckigen Symbolen. Für diese benötigen wir insgesamt 56 Positionsnummern. Bitte erinnern Sie sich, daß wir

pro Rechteck zwei Angaben brauchen, die nicht den eben erklärten Bildschirmpositionen entsprechen. In der Breite stehen uns doppelt so viele Adressen zur Verfügung (0 bis 127) und in der Höhe das Dreifache (0 bis 47). Das bedeutet, daß Sie sechsmal mehr Lichtsymbole als normale Zeichen auf dem Bildschirm darstellen können. Es ist müßig, die achtundzwanzig Lichblöcke mit ebenso vielen SET-Befehlen auf dem Bildschirm anzuzeigen. Wir verwenden hierzu die Kombination der DATA-Anweisung mit dem READ-Befehl:

```
1030 DATA 88,6,90,6,92,6,94,6,86,7,96,7
1040 DATA 84,8,98,8,82,9,100,9,81,10,101,10
1050 DATA 81,11,101,11,81,12,101,12,81,13,
     101,13
1060 DATA 82,14,100,14,84,15,98,15,86,16
1070 DATA 96,16,88,17,90,17,92,17,94,17
1080 FOR J = 1 TO 28
1090      READ H,V:SET (H,V)
1100 NEXT J
```

Wenn der Befehl in Zeile 1090 vom TRS-80 erstmalig durchgeführt wird, findet ein Lesevorgang statt, mit dem in den Variablen H und V die ersten beiden mit DATA festgelegten Werte gespeichert werden. Da dies die Zahlen 88 und 6 sind, lautet der Befehl in Zeile 1090 eigentlich SET (88,6). Beim nächsten Durchgang werden die Angaben 90 und 6 gelesen, und der SET-Befehl bezieht sich auf diese Zahlen. Mit der Variablen J wird festgelegt, daß die Schleife achtundzwanzigmal durchgeführt wird. Durch diesen Trick

werden alle DATA-Definitionen verarbeitet, und die Seifenblase ist fertig.
Im nächsten Schritt nimmt das Programm den Anfangsbuchstaben des Mitspielers auf und gibt ihn auf Position 326 aus:

```
1110 PRINT @576,"BITTE GEBEN SIE IHREN
     ANFANGSBUCHSTABEN EIN:"
1120 INPUT I$
1130 PRINT  @326,I$
```

Zeile 1010 enthält die Anweisung, den Text in der zehnten Zeile des Bildschirms auszugeben, Zeile 1130 besteht aus dem Befehl, zurück zur sechsten Bildschirmzeile zu springen und dort den Anfangsbuchstaben zu zeigen. Zu guter Letzt erklären wir dem Teilnehmer noch die Spielregel:

```
1140 PRINT @640,"BEANTWORTEN SIE EI-
     NE FRAGE RICHTIG, UND"
1150 PRINT "IHR ANFANGSBUCHSTABE
     BRINGT DIE SEIFENBLASE ZUM PLAT-
     ZEN."
1160 FOR J = 1 TO 500:NEXT J
1170 RETURN
```

Die Ausgabeposition 640 ist die elfte Zeile des Bildschirms, der Text wird also in Zeile 11 und 12 sichtbar. Im Programmbefehl 1160 geht es wieder darum, eine kurze Pause einzulegen, um dem Spieler Gelegenheit zu geben, den gesamten Text zu lesen, bevor es weitergeht.

Wie die Frage gestellt wird



In der Unterroutine 2000 werden dem Spieler so lange Fragen gestellt, bis er eine richtig beantwortet hat. Nach jeder Frage müssen wir einen Teil des Bildschirms löschen, und dafür benutzen wir eine zweite Unterroutine, die auf Zeile 2500 beginnt.

```
2000 REM————PROBLEMSTELLUNG——
     ————
2010 X = RND(100)
2020 Y = RND(100)
2070 GOSUB 2500:REM———LÖSCHUNG
     DER UNTEREN BILDSCHIRMHÄLF-
     TE—
2040 PRINT @576,"WIEVIEL IST ";X;" +
     ";Y;"?"
2050 INPUT A
```

```
2060 IF A <> X + Y GOTO 2010
2070 GOSUB 2500:REM——LÖSCHUNG
     DER UNTEREN BILDSCHIRMHÄLF-
     TE—
2080 RETURN
```

Zuerst greifen wir zwei Zufallszahlen zwischen 1 und 100 auf und nennen sie X und Y. Sie können natürlich die 100 in eine Ganzzahl in beliebiger Höhe umwandeln. Anschließend löschen wir die untere Bildschirmhälfte und zeigen die Additionsaufgabe auf der zehnten Zeile (Adresse 576). Dieser Vorgang wird wiederholt, bis eine richtige Lösung eingegeben wird. In diesem Moment (A = X + Y) verzweigt das Programm auf Befehlszeile 2060 und führt auch hier eine Löschung durch. Anschließend erfolgt der Rücksprung zum Hauptprogramm.
Aber wie wird gelöscht? Dies erfolgt einfach durch Positionierung auf jede Bildschirmposition und die Ausgabe einer Leerstelle mit PRINT:



```
2500 PRINT—————LÖSCHUNG DER UNTE-
     REN BILDSCHIRMHÄLFTE—
2510 FOR J = 512 TO 703
2520       PRINT @J," "
2530      NEXT J
2540 RETURN
```

Hinter den Adressen 512 bis 704 verbergen sich die Bildschirmzeilen 9 bis 11.

Die Seifenblase zum Platzen bringen



Nun besteht unsere Arbeit aus der Bewegung des Anfangsbuchstabens von seiner Ausgangsposition in Richtung auf die Seifenblase. Wir haben das Zeichen auf Stelle 326 ausgegeben, also löschen wir es hier erst einmal durch Ausgabe einer Leerstelle. Dann geben wir das Zeichen auf

Position 327 aus, um ihm den Anschein der Bewegung um eine Stelle zu geben. Hier wird es auch wieder gelöscht und wiederum eine Stelle weiter angezeigt. Diese Schritte werden so lange wiederholt, bis sich das Zeichen auf Adresse 363, auf der linken Seite der Seifenblase befindet.
Im Rennen der Anfangsbuchstaben mußten wir die Spur der beiden Rechtecke verfolgen und jedesmal überprüfen, ob eines der beiden durch seine Bewegung die Ziellinie überquert hatte. Wir konnten ohne diese Aufzeichnung nicht mit Sicherheit bestimmen, auf welcher Position sich ein Zeichen zu irgendeinem beliebigen Zeitpunkt befand, da der Ablauf durch eine Zufallszahl bestimmt wurde. Hier wissen wir nun ganz genau, wie wir das in I$ gespeicherte Initial bewegen wollen, so daß wir eine Schleife für die Bewegung aufbauen können, die das Zeichen von Position 326 nach Position 363 bringt:

```
3000 REM—————PLATZEN DER SEIFEN-
     BLASE————
3010 FOR H = 326 TO 363
3020      PRINT @H," "
3030      PRINT @H+1,I$
3040      FOR J = 1 TO 50:NEXT J
3050 NEXT H
```

Diese Programmschritte kennen wir schon: Auf jeder Adresse von 326 bis 362 löschen wir den Anfangsbuchstaben und geben ihn in der nächstfolgenden Spalte wieder aus. Im letzten

Schleifendurchgang enthält H den Wert 326, und das Zeichen wird in Spalte H + 1, das heißt auf 363, erscheinen.
Der Anfangsbuchstabe befindet sich nun links in der »Seifenblase«. Wenn wir den ganzen Ballon löschen, wird das Zeichen auch gelöscht. Wir könnten natürlich auf die gleiche Art löschen, wie wir die Zeichnung gemacht haben — durch die Benutzung von DATA für die RESET-Adressen —, aber diesmal durch die Ausgabe von Leerstellen auf jeder Position. Wir versuchen es aber mal mit einer anderen Methode. Wir wissen, daß die Blase eine bestimmte Fläche auf dem Schirm bedeckt, und zwar die Zeilen 3, 4, 5 und 6. Keine Zeile ist jedoch völlig ausgenutzt, nur elf Stellen sind belegt.

```
3060 FOR J = 168 TO 360 STEP 64
3070      PRINT @J,"           "
3080 NEXT J
```

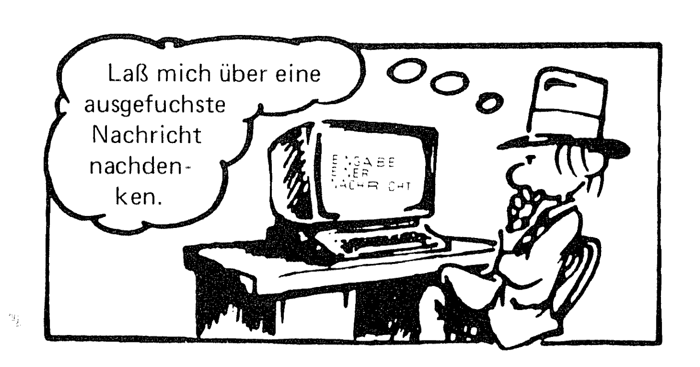
Wenn Sie nun einmal das Bildschirmformular ansehen, erkennen Sie, daß die Adresse 168 dem oberen linken Rand der Seifenblase entspricht. Ab dieser Position geben wir elf Leerstellen aus, unmittelbar darunter wiederum die gleiche Anzahl. Dieser Vorgang wird danach noch zweimal wiederholt. Wir wissen ja, daß jede Zeile aus 64 Stellen besteht, also wird die Schleife in entsprechenden Sprüngen ab dem Wert 168 wiederholt (Zeile 3060), und jedesmal werden elf Leerstellen ausgegeben.
Abschließend geben wir die Nachricht »RICHTIG!

GEWONNEN!« im Inneren der Blase aus — das heißt, dort, wo sie einmal war.

```
3090 PRINT @297,"RICHTIG! GEWONNEN!"
3100 RETURN
```

Die Adresse 297 befindet sich ungefähr in der Mitte der ehemaligen Seifenblase. Der Spieler wird den Eindruck haben, die Blase sei verschwunden oder »explodiert« und hätte den Text »RICHTIG! GEWONNEN!« enthalten.

Das ganze Programm



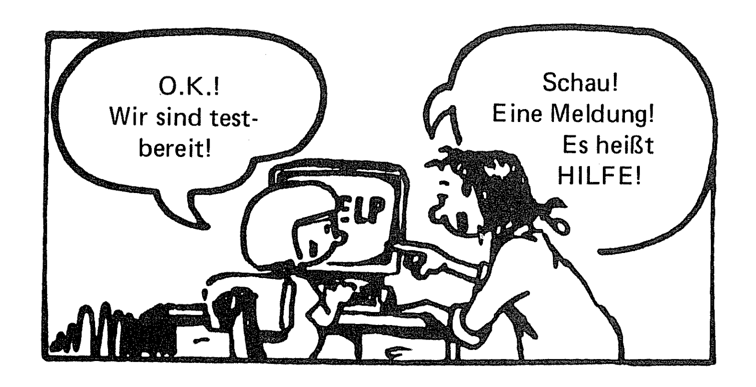

```
10 REM————DIE SEIFENBLASE
   PLATZT————— ——
20 REM I$     : ANFANGSBUCHSTABE
30 REM H      : HORIZONTALE POSITION
```

```
  40 REM V       : VERTIKALE POSITION
  50 REM X       : ZUFALLSZAHL FÜR DAS
     PROBLEM
  60 REM Y       : ZUFALLSZAHL FÜR DAS
     PROBLEM
  70 REM A       : ANTWORT
  80 REM J       : ZÄHLER
  90 REM————————
 100 GOSUB 1000:REM——BILDSCHIRM-
     AUFBAU——————
 200 GOSUB 2000:REM———PROBLEM (S)—
     ———
 300 GOSUB 3000:REM———PLATZEN DER
     SEIFENBLASE————
 400 END
1000 REM————BILDSCHIRMAUFBAU—
     ———
1010 CLS
1020 PRINT @20,"DIE SEIFENBLASE
     PLATZT"
1030 DATA 88,6,90,6,92,6,94,6,86,7,96,7
1040 DATA 84,8,98,8,82,9,100,9,81,10,101,10
1050 DATA 81,11,101,11,81,12,101,12,81,13,
     101,13
1060 DATA 82,14,100,14,84,15,98,15,86,16
1070 DATA 96,16,88,17,90,17,92,17,94,17
1080 FOR J = 1 TO 28
1090        READ H,V:SET (H,V)
1100 NEXT J
1110 PRINT @576,"BITTE GEBEN SIE IHREN
     ANFANGSBUCHSTABEN EIN:"
1120 INPUT I$
1130 PRINT  @326,I$
```

```
1140 PRINT  @640,"BEANTWORTEN SIE EI-
     NE FRAGE RICHTIG, UND"
1150 PRINT "IHR ANFANGSBUCHSTABE
     BRINGT DIE SEIFENBLASE ZUM PLAT-
     ZEN."
1160 FOR J = 1 TO 500:NEXT J
1170 RETURN
2000 REM————PROBLEMSTELLUNG——
     ———
2010 X = RND(100)
2020 Y = RND(100)
2030 GOSUB 2500:REM——LÖSCHUNG
     DER UNTEREN BILDSCHIRMHÄLF-
     TE—
2040 PRINT @576,"WIEVIEL IST ";X;" +
     ";Y;"?"
2050 INPUT A
2060 IF A <> X + Y GOTO 2010
2070 GOSUB 2500:REM——LÖSCHUNG
     DER UNTEREN BILDSCHIRMHÄLF-
     TE—
2080 RETURN
2500 REM———LÖSCHUNG DER UNTEREN
     BILDSCHIRMHÄLFTE—
2510 FOR J = 512 TO 703
2520        PRINT @J," "
2530       NEXT J
2540 RETURN
3000 REM————PLATZEN DER SEIFEN-
     BLASE——————
3010 FOR H = 326 TO 363
3020       PRINT @H," "
3030       PRINT @H+1,I$
```

```
3040       FOR J = 1 TO 50:NEXT J
3050 NEXT H
3060 FOR J = 168 TO 360 STEP 64
3070       PRINT @ J,"          "
3080 NEXT J
3090 PRINT @297,"RICHTIG! GEWONNEN!"
3100 RETURN
```

Variationen



Auch bei diesem Programm können Sie selbständig Änderungen vornehmen, zum Beispiel:
1. Stellen Sie anstatt einer Additions- eine Subtraktionsaufgabe. Die einzigen Zeilen, die verändert werden müssen, sind 2040 und 2060. Sie können die gleichen Zahlen, nämlich X und Y, benutzen. In 2040 formulieren Sie eine andere Fra-

ge und setzen das Minus- anstelle des Pluszeichens ein. In 2060 lassen Sie den Computer ermitteln, ob die Antwort (A) gleich oder nicht gleich X minus Y (X—Y) ist. Mit ähnlich einfachen Änderungen können Sie leicht auch Multiplikations- oder Divisionsfragen stellen.

2. Unser nächster Vorschlag ist verbunden mit einer etwas größeren Änderung. Fänden Sie es nicht toll, wenn bei jedem Durchlauf des Programms eine andere Nachricht »im Innern« der Seifenblase erscheinen würde? Sie wissen bereits, wie man das macht, denn wir haben schon mit dem Zufallsgenerator die Voraussagen und Meldungen gespeichert. Ganz am Ende der letzten Unterroutine, bei Zeile 3090, müssen Sie alles einfügen, was die Zufallsmeldungen ans Laufen bringt: DATA für die Meldungen, den Ladevorgang (READ) für den entsprechenden Tabellenbereich und die Auswahl der jeweiligen Nachricht durch den Zufallsgenerator für die Bildschirmausgabe. Sie könnten dafür natürlich auch eine eigene Unterroutine schreiben, wenn Sie glauben, die PLATZEN-Routine sei ohnehin schon zu lang. Das aber überlassen wir Ihnen.

6
Verwirrte Staaten

Alle Techniken, die wir bis jetzt gelernt haben, kombinieren wir nun mit einigen neuen, um ein noch umfangreicheres Spielprogramm herzustellen. In diesem Spiel geht es darum, mit vier Bewegungen vom Start zum Ziel zu gelangen. Die Bewegung findet aber nur dann statt, wenn der Spieler die richtige Antwort auf die Frage gibt, um welchen europäischen Staat es sich handelt. Der Name des Staates wird ihm in völlig verdrehter Form auf dem Bildschirm gezeigt, und der Spieler muß schon einiges an Kombina-



tionsfähigkeit aufbieten, um die Frage richtig zu beantworten. Wir geben dem Spieler insgesamt siebenmal die Chance, die vier Bewegungen aufgrund einer richtigen Eingabe durchzuführen. Wenn er oder sie damit die Ziellinie nicht erreicht, ist das Spiel beendet.
Die Namen der Variablen und die Liste der Unterroutinen sieht folgendermaßen aus:

```
10 REM------VERWIRRTE STAATEN--
   ---
20 REM I$     : ANFANGSBUCHSTABE DES
   SPIELERS
25 REM R$     : EINE ANTWORT
30 REM J      : EIN ZÄHLER
35 REM S$(10):TABELLE FÜR ZEHN LÄN-
   DERNAMEN
40 REM O$     : SPIELENDZEICHEN
45 REM T      : ANZAHL ERFOLGTER VER-
   SUCHE
50 REM N$     : DER ZU MISCHENDE LÄN-
   DERNAME
55 REM N      : ZUFALLSZAHL FÜR STAA-
   TEN
60 REM L      : LÄNGE DES NAMENS
65 REM U$(30):TABELLE DER BUCH-
   STABEN IM NAMEN
70 REM R      : ZUFALLSZAHL FÜR
   BUCHSTABEN
75 REM P      : POSITION DES INITIALS
80 REM M      : ÜBERTRAG
90 REM------
```

```
100 GOSUB 1000:REM———BILDSCHIRM-
    AUFBAU—————
200 GOSUB 2000:REM———STAATENTA-
    BELLE—————
300 GOSUB 3000:REM———SPIEL————
400 END
```

Lassen Sie sich von den dreizehn Variablennamen im gleichen Programm nicht aus der Fassung bringen. Es ist für Sie sicher einfacher, sich zunächst mit den einzelnen Unterroutinen und den dort benötigten Variablen vertraut zu machen, denn pro Unterroutine werden wir nur einige wenige verwenden.
Zunächst sehen wir im Hauptprogramm nur drei Unterroutinen, aber wenn wir an der SPIEL-Unterroutine angelangt sind, werden Sie merken, daß diese wiederum drei andere Unterroutinen aufruft. Innerhalb des Spieles müssen wir den Bildschirm löschen, eine Frage stellen und einen Übertrag, das heißt eine Bewegung, durchführen. Und diese drei Dinge müssen wir ständig bis zu einem bestimmten Punkt wiederholen. Deshalb ist jede dieser Aktivitäten in einer eigenständigen Unterroutine enthalten.
Den Ablauf bis zur SPIEL-Unterroutine sollten wir kennen, also sollten wir auch in der Lage sein, die ersten beiden Abteilungen problemlos zu durchlaufen.

Der Bildschirmaufbau

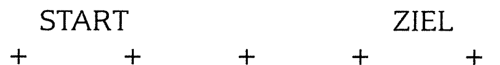


Wir beginnen mit der Spielvorbereitung, indem wir auf dem Bildschirm vier Positionen markieren, die den jeweiligen Startpunkt für die vier Bewegungen darstellen, die der Anfangsbuchstabe des Spielers machen muß, um die Ziellinie zu erreichen. Auch in diesem Fall bleibt es Ihnen überlassen, den Bildschirm durch ein ausgetüfteltes »Spielbrett« zu verschönern. Sie können leicht alle möglichen dekorativen Elemente auf dem Bildschirm darstellen. Wir beschränken uns hier auf die Markierung der drei Startpositionen durch Pluszeichen (+), positionieren den Anfangsbuchstaben am Startpunkt der Bewegungslinie und erklären den Spielverlauf:

```
1000 REM———BILDSCHIRMAUFBAU——
     ————
```

```
1010 CLS
1020 PRINT @200,"START"
1030 PRINT @240,"ZIEL"
1040 FOR J = 266 TO 306 STEP 10
1050      PRINT @J,"+"
1060 NEXT J
```

Nach dieser Befehlsfolge ist eigentlich die meiste Arbeit in der Unterroutine schon getan, die Bildschirmzeile 4 enthält die Texte START und ZIEL, die 5. Zeile fünf Pluszeichen auf den Adressen 266, 276, 286, 296 und 306. Da wir sie alle zehn Positionen ausgeben, benutzen wir in Programmzeile 1040 die Angabe STEP 10. Die Zeilen 4 und 5 des Bildschirms sehen jetzt so aus:



Nun wird der Anfangsbuchstabe des Spielernamens auf dem ersten Pluszeichen positioniert:

```
1070 PRINT @512,"GEBEN SIE BITTE IHREN
     ANFANGSBUCHSTABEN EIN:"
1080 INPUT I$
1090 PRINT @266,I$
```

Die letzte Adresse entspricht dem + unter dem Wort START, das nach der Ausgabe des Initials nun nicht mehr sichtbar ist.
Am Ende der Unterroutine geben wir jetzt noch eine Erklärung des Spielablaufs beziehungsweise der Regeln auf dem Bildschirm aus:

```
1100 PRINT @512,"SIE HABEN 6 VERSU-
     CHE, UM ANS ZIEL ZU GELANGEN."
1110 PRINT "UM SICH ZU BEWEGEN, MÜS-
     SEN SIE DEN"
1120 PRINT "NAMEN EINES STAATES IN
     EUROPA ENTZIFFERN."
1130 PRINT @768,"DRÜCKEN SIE ZUM
     START DIE RETURN-TASTE"
1140 INPUT R$
1150 RETURN
```

Nach Drücken der RETURN-Taste wird der Computer zum Hauptprogramm zurückkehren und mit der Abarbeitung der nächsten Unterroutine beginnen.

Die Speicherung der Staaten



Können Sie sich noch an die Unterroutine in Kapitel 1 erinnern, mit der wir mehrere Zukunftsvoraussagen in einer Tabelle gespeichert haben? In diesem Fall handelt es sich um zehn Staaten, deren Namen wir in DATA-Anweisungen angeben. Beachten Sie bitte, daß pro DATA-Zeile mehrere Dateneinträge, das heißt Ländernamen, dargestellt werden können, solange die einzelnen Zeichenketten durch Kommata getrennt werden. Wir nehmen auch hier den READ-Befehl, um die Ländernamen in eine Tabelle zu laden, die wir mit S$ (S als Abkürzung für Staat) bezeichnet haben. Bevor wir diese Tabelle jedoch benutzen können, müssen wir sie DIMensionieren — das heißt, der Computer wird angewiesen, zehn Leerbereiche für unsere Tabelle zu reservieren.

```
2000 REM———STAATENTABELLE—————
2010 DATA "BUNDESREPUBLIK","ENG-
     LAND","FRANKREICH","HOLLAND",
     "LUXEMBURG","BELGIEN","SPANIEN"
2020 DATA "ITALIEN","GRIECHENLAND"
2030 DATA "ÖSTERREICH"
2040 DIM S$(10)
2050 FOR J = 1 TO 10
2060      READ S$(J)
2070 NEXT J
2080 RETURN
```

Die Schleife in den Zeilen 2050 und 2070 bewirkt zehn Lesevorgänge (READs) zur Speicherung der zehn Ländernamen in unserer Tabelle. Der

Name »BUNDESREPUBLIK« wird damit auf Platz 1 der Tabelle, also S$(1), der Name »FRANKREICH« auf S$(3) und der Name »ÖSTERREICH« auf S$(10) gespeichert. Im weiteren Verlauf des Programms werden wir diese Tabelle in derselben Weise verwenden wie die Weissagungstabelle. Wir greifen eine Zufallszahl auf und benutzen diese zur Auswahl einer der Datenzellen. Anschließend wird der in dieser Zelle gespeicherte Ländername herausgenommen und durcheinandergeschüttelt.

Das eigentliche Spiel



Wir haben vorhin festgelegt, daß der Spieler sieben Möglichkeiten erhält, die Ziellinie zu erreichen. Jede Runde besteht aus dem Versuch, einen zufällig angezeigten Ländernamen zu ent-

ziffern, das heißt, die Buchstaben in die richtige Reihenfolge zu bringen. Wir müssen unsere Aufmerksamkeit also noch auf weitere Dinge richten:

1. Wo sich der Anfangsbuchstabe zur Zeit befindet,

2. ob der Anfangsbuchstabe inzwischen die Ziellinie erreicht hat,

3. wie viele Versuche der Spieler bereits hinter sich hat.

Die SPIEL-Unterroutine ist ziemlich komplex. Sie liest sich wie ein eigenständiges kleines Programm und enthält selbst die Verzweigung auf weitere Unterroutinen. Bevor wir uns das in BASIC anschauen, wollen wir in allgemeinverständlicher Sprache die Schritte verdeutlichen, die wir durchführen müssen:

Überprüfung, ob der Spieler schon gewonnen hat ODER ob die maximal mögliche Anzahl der Versuche überschritten ist.

Wenn eine dieser beiden Bedingungen zutrifft, geben wir die entsprechende Meldung aus.

Wenn keine von beiden erfüllt ist, führen wir folgende Aktionen durch:

1. Löschung des unteren Bildschirmteils,

2. per Zufallsgenerator einen Ländernamen aufgreifen,

3. die Buchstaben des ausgewählten Ländernamens mischen,

4. den Spieler zur Entschlüsselung oder Richtigstellung auffordern,

5. ermitteln, ob die eingegebene Antwort richtig oder falsch ist.

a) Wenn sie richtig ist, den Anfangsbuchstaben um zehn Stellen weiterbewegen und überprüfen, ob der Spieler nun gewonnen hat.
b) Wenn sie NICHT stimmt, dem Spieler den richtigen Ländernamen mitteilen.
c) In jedem Fall addieren wir eine 1 zu der Zahl der bereits erfolgten Versuche und kehren zurück zum ersten Schritt.
(Überprüfung, ob der Spieler gewonnen hat...)

Die Zeilen 3030 bis 3150 sind die gleichen Schritte in BASIC. Vor diesen Schritten DIMensionieren wir eine Tabelle U$, die wir für das Mischen der Buchstaben des Ländernamens benötigen. Anschließend setzen oder initialisieren wir drei Variable: T steht für die Anzahl Versuche und wird zu Beginn des Spiels auf 1 gesetzt, da wir uns ja im ersten Versuch befinden. P steht für Position und stellt die Adresse des Anfangsbuchstabens des Spielers dar. Erinnern Sie sich? Wir setzten das Initial zu Beginn auf 266. O$ steht für Spielende, und wir benutzen die Variable zur Kennzeichnung, ob das Spiel nun beendet ist oder nicht — sie wird mit »NEIN« initialisiert.

```
3000 REM————SPIEL———————
3010 DIM U$(30)
3020 T = 1:P = 266:O$ = "NEIN"
3030 IF T > 7 OR O$ = "JA" GOTO 3160
3040     GOSUB 3700:REM——LÖSCHUNG
         DES UNTEREN BILDSCHIRMTEILS—
3050     GOSUB 3300:REM——FRAGESTEL-
         LUNG
```

```
3060 PRINT @704,"BITTE EINGABE DER
     ANTWORT:"
3070   INPUT A$
3080   IF A$ = N$ GOTO 3120
3090        PRINT @512,"SCHADE! DER
     STAAT IST ";N$
3100        FOR J = 1 TO 100:NEXT J
3110        GOTO 3140
3120        PRINT @832,"RICHTIG!"
3130        GOSUB 3500:REM———ÜBER-
     TRAG
3140   T = T + 1
3150   GOTO 3030
3160 GOSUB 3700:REM———LÖSCHUNG
     DES UNTEREN BILDSCHIRMTEILS
3170 IF O$ = "JA" GOTO 3200
3180   PRINT @512," SCHADE, DIE VERSU-
     CHE SIND BEENDET!"
3190   GOTO 3210
3200   PRINT @512," GLÜCKWUNSCH! SIE
     HABEN GEWONNEN!"
3210 RETURN
```

In Zeile 3030 programmieren wir ZWEI Bedingungen, von denen nur eine jeweils zutreffen kann; das Spiel ist vorbei, wenn die maximale Versuchsanzahl erreicht ist ODER wenn der Spieler gewonnen hat. Das Programm erkennt, daß das Spiel gewonnen wurde, wenn das Wort »JA« in O$ gespeichert ist. In Zeile 3020 haben wir in O$ das Wort »NEIN« hinterlegt. Wie kommt nun das JA hinein? Jedesmal, wenn wir den Anfangsbuchstaben bewegen, prüfen wir, ob er die Ziel-

linie erreicht hat, und wenn das zutrifft, ändern wir den Inhalt von O$ in JA. Das erfolgt in der ÜBERTRAG-Unterroutine, die wir kurz untersuchen wollen.
Wenn das Spiel noch nicht beendet ist, verzweigt das Programm auf Zeile 3040. Nun erwarten wir vom Programm die Löschung des unteren Bildschirmbereichs wie im letzten Kapitel. Anschließend wird eine Frage gestellt. Beide Arbeitsabläufe sind groß genug, um eigenständige Unterroutinen darzustellen. In der FRAGESTELLUNG-Unterroutine greifen wir den Namen des Landes auf und nennen den Speicherbereich N$. Deshalb überprüfen wir auch die Eingabe im Speicher A$ auf Identität mit dem Inhalt von N$. Wenn beide Bereiche übereinstimmen, springt das Programm zur Zeile 3120 und gibt den Text RICHTIG! in der vierzehnten Zeile aus, daraufhin wird das Initial bewegt. Ist die Antwort aber falsch, wird dem Spieler der eigentliche Ländername mitgeteilt, und das Initial bleibt an seinem Platz.
Am Ende jedes Schleifendurchlaufs addieren wir 1 zum Inhalt von T (Zeile 3140) und gehen zurück zur Zeile 3030, um zu überprüfen, ob das Spiel nun beendet ist. Früher oder später wird T einen Wert größer als 7 annehmen. In diesem Fall wird das Wort »JA« in O$ gespeichert. Dann springt das Programm zur Zeile 3160, wo wiederum der untere Bildschirmbereich gelöscht wird. Dann wird überprüft, aus welchem Grund das Spiel beendet wurde. Hat der Spieler gewonnen, wird auf Zeile 3200 verzweigt und eine Gratulations-

zeile ausgegeben, andernfalls teilt der Computer sein Bedauern mit, daß das Spiel verloren wurde.

Das Löschen des unteren Bildschirmbereichs



Jetzt ist es an der Zeit, daß wir mit der Untersuchung der drei Unterroutinen innerhalb der SPIEL-Unterroutine beginnen. Jedesmal, wenn wir eine neue Frage stellen oder eine neue Meldung ausgeben, müssen wir den unteren Teil des Bildschirms löschen. Im Gegensatz zu Kapitel 5 löschen wir hier jedoch einen größeren Bereich, nämlich von Adresse 512 bis 959. Und so wird's gemacht:

```
3700 REM———LÖSCHUNG DES UNTEREN
          BILDSCHIRMTEILS—
```

```
3710 FOR J = 512 TO 959
3720      PRINT @J," "
3730 NEXT J
3740 RETURN
```

Für diese Aktion benötigt der Computer schon eine gewisse Zeit, da wir ihm zumuten, jede einzelne Stelle zwischen 512 und 959 mit einen Leerwert zu versehen. Diesen Befehl führt er nämlich auf jeden Fall durch, egal, ob auf dieser Adresse ein Zeichen steht oder nicht.

Das Durcheinanderwirbeln des Ländernamens



Wir haben eine Tabelle bestehend aus zehn Staaten, die im Programm mit dem Symbol S$(10) be-

zeichnet ist. Immer, wenn der Zeitpunkt für eine erneute Fragestellung gekommen ist, wollen wir einen Ländernamen aufgreifen und die Zeichenkette durcheinanderbringen. Das erste, was bekannt sein muß, ist die Anzahl der Buchstaben des herausgegriffenen Ländernamens, den wir mit N$ bezeichnet haben. Die LEN-Funktion, so erinnern wir uns, gibt uns über die Gesamtzahl der Zeichen in einer Kette Auskunft. Also stellt LEN(W$) die Anzahl der Zeichen des aktuellen Ländernamens dar. Wir reden von Zeichen und nicht von Buchstaben, da die Namen ja auch Bindestriche oder Leerstellen enthalten können. Die Zeichenanzahl im Ländernamen markieren wir mit dem Variablennamen L. Um sie zu mischen, müssen wir eine Serie von Schritten auslösen:

1. Aufgreifen einer Zufallsganzzahl (genannt R) zwischen 1 und L,

2. Ausgabe des Zeichens mit der Positionsnummer R auf dem Bildschirm,

3. Aufgreifen einer weiteren Zufallsganzzahl R zwischen 1 und L,

4. Ausgabe eines weiteren Zeichens mit der Positionsnummer R als Folgezeichen unmittelbar hinter dem bereits angezeigten Zeichen.

Werden die oben angeführten Schritte insgesamt L-mal wiederholt, dann erscheinen L-Zeichen auf dem Bildschirm. Diese Schrittfolge enthält ein Problem, auf das wir Sie noch nicht aufmerksam gemacht haben: Sie müssen wissen, daß der Computer bei der Bereitstellung einer Zufallszahl zwischen 1 und 7 beispielsweise

mehrfach dieselbe Zahl angeben kann. Wenn also HOLLAND der zu erratende Staat ist, und der Computer gibt ständig die 4 an, würde die gemixte Ausgabe von HOLLAND aussehen wie »LLLLLLL«. Und das würde natürlich den weiteren Spielverlauf erheblich behindern.
Aus diesem Grund muß ein Weg gefunden werden, den Computer anzuweisen, die Zufallszahl aus dem Bereich 1 bis 7 zu bilden, aber keine Zahl zweimal zu verwenden. Mit anderen Worten: Wir müssen die schon verwendeten Zahlen aufzeichnen. Die beste Art, das zu tun, besteht aus der Festlegung einer weiteren Tabelle, die wir mit U$ bezeichnen. Bevor wir nun mit dem Mix beginnen, speichern wir in jeder Zelle von U$ das Wort »NEIN« ab. Es zeigt uns an, daß noch keine Zahl benutzt wurde. Wenn nun eine Zahl aufgegriffen wurde, speichern wir in der zugeordneten Zelle das Wort »JA« ab. Übrigens DIMensionieren wir die Tabelle zu Beginn der SPIEL-Unterroutine mit dem Wert 30. Nun wollen wir unsere Schrittfolge diesen Überlegungen anpassen und sehen, wie sich diese Ergänzung auswirkt:
1. Aufgreifen einer Zufallszahl zwischen 1 und L,
2. Überprüfen, ob diese Zahl schon verwendet wurde:
a) wenn ja, zurück zu Schritt 1.
b) wenn nein, Speicherung von »JA« in der Zelle von U$, die der Zufallszahl entspricht, und Ausgabe des entsprechenden Zeichens aus dem Ländernamen.
Wiederholung der Schrittfolge L-mal.
Der ganze Bereich sieht dann in BASIC so aus:

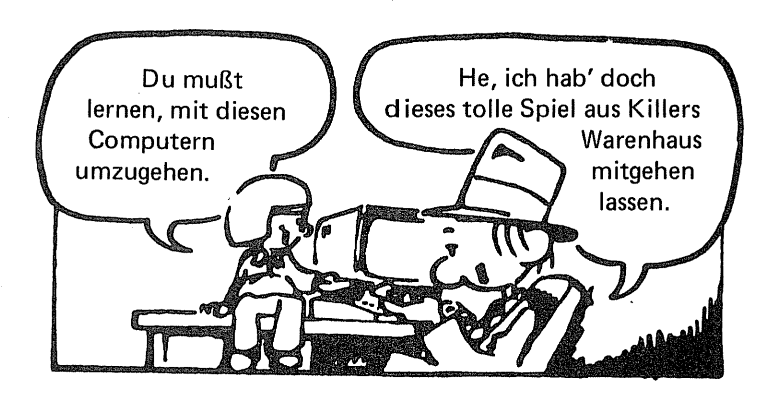
```
3300 REM———FRAGESTELLUNG———
     ——
3310 N = RND(10)
3320 N$ = S$(N)
3330 L = LEN(N$)
3340 FOR J = 1 TO L
3350      U$(J) = "NEIN"
3360 NEXT J
3370 PRINT @512,"WELCHER STAAT IST
     DAS?"
3380 FOR J = 1 TO L
3390      R = RND(L)
3400      IF U$(R) = "JA" GOTO 3390
3410      U$(R) = "JA"
3420      PRINT @J+580,MID$(N$,R,1)
3430 NEXT J
3440 RETURN
```

N ist eine Zufallsganzzahl von 1 bis 10. Warum 10? Weil wir genau zehn Staaten haben. Wenn N eine 1 enthält, dann ist S$(N) gleich BUNDESREPUBLIK, also der erste Staat in der DATA-Auflistung. Im Programm leichter verständlich ist die Speicherung von S$(N) in N$. Wenn N$ den Namen FRANKREICH enthält, hat L den Wert 10, nämlich die Anzahl aller Zeichen des Wortes FRANKREICH.
Die Schleife in den Zeilen 3340 bis 3360 initialisiert die Tabelle U$. Sie bewirkt den Eintrag von »NEIN« in den 10 Zellen von U$. Also enthalten U$(1), U$(2), U$(3) und so weiter bis U$(10) zu Beginn das Wort »NEIN«. Um bei unserem Beispiel zu bleiben: Das Wort FRANKREICH besteht

aus zehn Zeichen, und sie alle wurden noch NICHT benutzt.
Nach der Frage WELCHER STAAT IST DAS? gelangen wir in den Teil der Unterroutine, der tatsächlich die Mischung der Zeichenfolge vornimmt und diese dann in verdrehter Reihenfolge ausgibt. Die FOR/NEXT-Schleife in den Zeilen 3380 bis 3430 wird L-mal durchgeführt. In unserem Beispiel mit FRANKREICH also zehnmal — aufgrund der zehn Buchstaben.
Jedesmal, wenn nun eine Zufallszahl von 1 bis 10 (L) aufgegriffen wird, erfolgt also die Prüfung, ob die entsprechende Zelle von U$ schon ein »JA« enthält, weil die Zahl bereits verwendet wurde. In diesem Fall kehrt das Programm zur Auswahl zurück, bis es eine Zahl findet, in deren zugeordneter Zelle das Wort »NEIN« enthalten ist. Daraufhin wird es mit der Verarbeitung zweier Anweisungen fortfahren. Erstens wird in der Zelle ein »JA« eingetragen, so daß derselbe Buchstabe nicht zweimal durch den Computer ausgegeben wird. Zweitens wird durch die Benutzung der MID$-Funktion das der Zufallszahl zugeordnete Zeichen aus N$ ausgegeben.
Die Zeile 3420 sollten wir genauer betrachten. Wie wir wissen, entnimmt die MID$-Funktion eine bestimmte Anzahl Zeichen aus einem gegebenen Textbereich. R ist nun die laufende Nummer des Zeichens in der Kette, das wir bearbeiten wollen. Somit bewirkt MID$(N$,R,1) die Festlegung und weitere Verarbeitung genau eines Zeichens aus N$. Wenn R den Wert 3 annimmt, wird der Computer den Buchstaben A aus dem

Wort FRANKREICH anzeigen, da dieser die laufende Nummer 3 trägt. Die Ausgabe soll auf Position 581 erfolgen, die des nächsten Zeichens auf 582 und so weiter: Deshalb weisen wir den Computer an, auf J+580 das Symbol sichtbar zu machen. Enthält J die Zahl 1, ist die Adresse 581. Im nächsten Durchlauf entspricht J der Zahl 2, also haben wir es mit Position 582 zu tun.
Es wäre sinnvoll, wenn Sie an dieser Stelle die einzelnen Schritte der Unterroutine mehrmals für sich nachvollziehen würden, um zu sehen, was jeder Schritt im einzelnen bewirkt. Vergegenwärtigen Sie sich, daß bei jedem Aufruf dieser Routine ein neuer Ländername, hier N$ genannt, durch den mehrmaligen Durchlauf einer Schleife per Zufallsgenerator gemischt und verkehrt ausgegeben wird. Diese Unterroutine dient nur diesem einen Zweck. Blättern Sie noch einmal zur eigentlichen SPIEL-Unterroutine zurück. Genau vor dem Aufruf der FRAGESTELLUNG löschen wir den unteren Bildschirmbereich. Unmittelbar danach gibt der Spieler seine Antwort ein, die dann auf Richtigkeit überprüft wird. Wenn diese stimmt, müssen wir mit einem weiteren Ablauf beginnen.

Die Bewegung des Anfangsbuchstabens



Jede richtige Antwort bringt den Anfangsbuchstaben dem Ziel einen Schritt näher. Wir haben Pluszeichen zur Markierung dieser Schritte auf dem Bildschirm ausgegeben, und zwar in einem Abstand von 10 Einheiten auf den Positionen 276, 286, 296 und 306 (dem Ziel). Erinnern wir uns auch daran, daß wir zu Beginn der SPIEL-Unterroutine die Variable P mit 266 initialisiert haben, um die Startposition des Anfangsbuchstabens aufzuzeichnen.
Zur Bewegung des Initials durchläuft unser Programm die gleichen Basisschritte, die wir in den letzten beiden Programmen schon verwendet haben:

1. Löschung des Zeichens auf seiner zuletzt erreichten Position,

2. Ausgabe des Zeichens auf der nächsten Position.

Wir wiederholen diese Schritte zehnmal, um den Anfangsbuchstaben um zehn Spalten vorwärts zu bewegen. Dann tun wir zwei weitere wichtige Dinge. Wir addieren die Zahl 10 auf P, um die neue Position des Zeichens festzuhalten, und wir prüfen, ob P den Wert 306 erreicht hat. Falls das zutrifft, heißt das, der Spieler hat gewonnen, und wir speichern das Wort »JA« in der Variablen O$.

```
3500 REM————ÜBERTRAG——————
3510 FOR M = P TO P + 9
3520        PRINT @M," "
3530        PRINT @M+1,I$
3540        FOR J = 1 TO 100:NEXT J
3550 NEXT M
3560 P = P + 10
3570 IF P = 306 THEN O$ = "JA"
3580 RETURN
```

M bewegt sich also von P bis P + 9. Warum nicht P + 10? Weil wir jedesmal I$ auf der Position M + 1 ausgeben. Bei der letzten Schleifendurchführung, wenn M dem Wert P + 9 entspricht, geben wir I$ auf M + 1 oder auch P + 10 aus.

O$ wurde am Anfang des Spiels auf den Wert »NEIN« gesetzt. Sobald P den Wert 306 annimmt, wird in O$ ein »JA« gespeichert. Zurück in der SPIEL-Abteilung, trifft nun die Bedingung in Zeile 3030 zu. Dadurch wird das Programm die nächste Schleife überspringen. Wenn es in der Zeile

3170 anlangt, wird der Text »GRATULATION!. . .« ausgegeben, da O$ dem Wort »JA« entspricht, und das Spiel ist somit beendet.

Das gesamte Programm

Das Spiel »Verwirrte Staaten« ist bei weitem das längste, das wir geschrieben haben. Aber es wird nichts benutzt, was Sie nicht schon kennen. Und jetzt wollen wir Ihnen nochmals das gesamte Programm vorstellen:

```
10 REM————VERWIRRTE STAATEN—
   ————
20 REM I$      : ANFANGSBUCHSTABE
   DES SPIELERS
25 REM R$      : EINE ANTWORT
30 REM J       : EIN ZÄHLER
35 REM S$(10):TABELLE FÜR ZEHN LÄN-
   DERNAMEN
40 REM O$      : SPIELENDZEICHEN
45 REM T       : ANZAHL ERFOLGTER
   VERSUCHE
50 REM N$      : DER ZU MISCHENDE
   LÄNDERNAME
55 REM N       : ZUFALLSZAHL FÜR STAA-
   TEN
60 REM L       : LÄNGE DES NAMENS
65 REM U$(30):TABELLE DER BUCH-
   STABEN IM NAMEN
70 REM R       : ZUFALLSZAHL FÜR
   BUCHSTABEN
```

```
  75 REM P      : POSITION DES INTIALS
  80 REM M      : ÜBERTRAG
  90 REM————————
 100 GOSUB 1000:REM———BILDSCHIRM-
     AUFBAU——————
 200 GOSUB 2000:REM———STAATENTA-
     BELLE——————
 300 GOSUB 3000:REM———SPIEL———
     ———
 400 END
1000 REM————BILDSCHIRMAUFBAU——
     —————
1010 CLS
1020 PRINT @200,"START"
1030 PRINT @240,"ZIEL"
1040 FOR J = 266 TO 306 STEP 10
1050      PRINT @J,"+"
1060 NEXT J
1070 PRINT @512,"GEBEN SIE BITTE IHREN
     ANFANGSBUCHSTABEN EIN:"
1080 INPUT I$
1090 PRINT @266,I$
1100 PRINT @512,"SIE HABEN 6 VERSU-
     CHE, UM ANS ZIEL ZU GELANGEN."
1110 PRINT "UM SICH ZU BEWEGEN, MÜS-
     SEN SIE DEN"
1120 PRINT "NAMEN EINES STAATES IN
     EUROPA ENTZIFFERN."
1130 PRINT @768,"DRÜCKEN SIE ZUM
     START DIE RETURN-TASTE"
1140 INPUT R$
1150 RETURN
2000 REM———STAATENTABELLE———
```

```
2010 DATA "BUNDESREPUBLIK","ENG-
     LAND","FRANKREICH","HOLLAND",
     "LUXEMBURG","BELGIEN","SPANIEN"
2020 DATA "ITALIEN","GRIECHENLAND"
2030 DATA "ÖSTERREICH"
2040 DIM S$(10)
2050 FOR J = 1 TO 10
2060      READ S$(J)
2070 NEXT J
2080 RETURN
3000 REM———SPIEL——————
3010 DIM U$(30)
3020 T = 1:P = 266:O$ = "NEIN"
3030 IF T > 7 OR O$ = "JA" GOTO 3160
3040    GOSUB 3700:REM——LÖSCHUNG
     DES UNTEREN BILDSCHIRMTEILS—
3050    GOSUB 3300:REM——FRAGESTEL-
     LUNG
3060    PRINT @704,"BITTE EINGABE DER
     ANTWORT:"
3070    INPUT A$
3080    IF A$ = N$ GOTO 3120
3090         PRINT @512,"SCHADE! DER
     STAAT IST";N$
3100         FOR J = 1  TO 100:NEXT J
3110         GOTO 3140
3120         PRINT @832,"RICHTIG!"
3130         GOSUB 3500:REM——ÜBER-
     TRAG
3140    T = T + 1
3150    GOTO 3030
3160 GOSUB 3700:REM——LÖSCHUNG
     DES UNTEREN BILDSCHIRMTEILS
```

```
3170 IF O$ = "JA" GOTO 3200
3180     PRINT @512,"SCHADE, DIE VERSU-
       CHE SIND BEENDET!"
3190     GOTO 3210
3200     PRINT @512,"GLÜCKWUNSCH! SIE
       HABEN GEWONNEN!"
3210 RETURN
3300 REM———FRAGESTELLUNG———
       ——
3310 N = RND(10)
3320 N$ = S$(N)
3330 L = LEN(N$)
3340 FOR J = 1 TO L
3350       U$(J) = "NEIN"
3360 NEXT J
3370 PRINT @512,"WELCHER STAAT IST
       DAS?"
3380 FOR J = 1 TO L
3390       R = RND(L)
3400       IF U$(R) = "JA" GOTO 3390
3410       U$(R) = "JA"
3420       PRINT @J+580,MID$(N$,R,1)
3430 NEXT J
3440 RETURN
3500 REM————ÜBERTRAG————————
3510 FOR M = P TO P + 9
3520       PRINT @M," "
3530       PRINT @M+1,I$
3540       FOR J = 1 TO 100:NEXT J
3550 NEXT M
3560 P = P + 10
3570 IF P = 306 THEN O$ = "JA"
3580 RETURN
```

```
3700 REM——LÖSCHUNG DES UNTEREN
     BILDSCHIRMTEILS——
3710 FOR J = 512 TO 959
3720      PRINT @ J," "
3730 NEXT J
3740 RETURN
```

Spaß und Spiele



An dieser Stelle liegen nun alle Variationen bei Ihnen. In den ersten fünf Kapiteln haben wir uns noch erlaubt, einige Verbesserungen vorzuschlagen. Sie können fast alle Hinweise auch in diesem Spiel verwirklichen, wenn Sie wollen. Aber wichtiger für Sie ist, sich einige Möglichkeiten der Verbesserung auszudenken und in die Tat umzusetzen. Gleiches gilt für die Erfindung weiterer Spiele. Entwerfen Sie Ihre eigenen Codierungsroutinen. Erfinden Sie weitere Wege, die Random-Funktion auszunutzen. Gestalten Sie die Animation nach eigenem Geschmack so phantasievoll, wie Sie nur können.
Programmierung ist eine Kombination aus der Benutzung dessen, was Sie bereits erlernt haben und beherrschen und der Nutzung der eigenen Vorstellungskraft. Wir haben versucht, Sie mit ei-

nigen wesentlichen Abläufen und Techniken vertraut zu machen. Natürlich müssen Sie immer noch eine Menge dazulernen — das gilt sogar für die erfahrensten Programmierer. Wir hoffen, daß Sie wenigstens einen Teil dessen, was wir Ihnen vorgeführt haben, zur Erfindung weiterer spannender Spiele verwenden können und daß wir Ihnen einen Anreiz zum zukünftigen Ausbau Ihrer Kenntnisse bieten konnten. Das Wichtigste, was Sie dazu beitragen können, ist der eigene Spaß an der Programmierung.

Kleines Lexikon und Index

Ein Verzeichnis der in diesem Buch behandelten neuen Abläufe und Fachausdrücke mit Querverweisen auf die Kapitel, in denen sie angesprochen werden. In BASIC reservierte Begriffe sind in Großbuchstaben dargestellt.

NAME	*BEDEUTUNG UND VERWENDUNG*

ARRAY
: deutsch: Tabelle, ein Variablentyp, in dem mehr als ein Datenelement gespeichert und unter Verwendung desselben Namens verarbeitet wird. (1,3,6)

Animation
: bewirkt die scheinbare »Bewegung« irgendeines Objekts, beispielsweise eines Zeichens über den Bildschirm durch schnelles Ausgeben und Löschen auf aufeinanderfolgenden Bildschirmpositionen. (4 — 6)

ASC-Funktion

setzt ein gegebenes Zeichen in die ASCII-Codezahl um. Beispiel: PRINT ASC ("A") bewirkt die Ausgabe der Codezahl 65. (3)

CHR$-Funktion

bewirkt die Umsetzung einer gegebenen ASCII-Codezahl in das zugeordnete Zeichen. Beispiel: PRINT CHR$(65) bewirkt die Ausgabe des Buchstabens A. (3)

CLEAR

reserviert eine festzulegende Anzahl Speicherstellen für Zeichen in einem Programm. (3)

DIM

ein Kommando, das eine gegebene Anzahl von Datenzellen für eine Tabelle reserviert. Beispiel: DIM M$(15) legt 15 Datenzellen im Speicher für die Tabelle M$ fest. (1,3,6)

FOR/NEXT

FOR und NEXT werden am Anfang und am Ende einer Wiederholungsschleife plaziert. (1 — 6)

Rückwärtszählung durch Benutzung negativer Schritte. (2)

Leere Schleifen für die Programmpause. (4 — 6)

GOSUB

(siehe Unterroutine)

Initialisierung
: die Festlegung und Speicherung von Anfangs- oder Initialisierungswerten. (1 — 6)

INT-Funktion
: rundet jede Zahl auf die nächstniedrigere Ganzzahl ab. Wird in Verbindung mit der RND(1)-Funktion angewendet. (1 — 6)

LEN-Funktion
: stellt die Gesamtanzahl von Zeichen in einer gegebenen Zeichenkette bereit. Beispiel: LEN("HALLO") ergibt eine 5. (2 — 3)

MID$-Funktion
: bewirkt die Wiedergabe einer definierten Zeichenkette innerhalb einer gegebenen Zeichenkette. Beispiel: MID$("TRS-80",5,2) beginnt beim fünften Zeichen, gibt also PC wieder. (2 — 3)

ON/GOSUB
: unterstützt ein Auswahlverfahren für die Durchführung von Unterroutinen. (3)

PRINT @
: zeigt einen Text ab einer bestimmten Bildschirmpositionsadresse an, die nach dem @-Zeichen beziffert wird. (5 — 6)

random
: Die RND-Funktion wählt Zufallszahlen, wie es in Kapitel 1 erklärt wurde, aus.

Wir haben diese Funktion wie folgt genutzt:
Zufallszahlen in Tabellenbereichszellen. (1,3,6)
Zufallszeichenzahlen in Verbindung mit der CHR$-Funktion. (3)
Zufallszahl für ein ON/GOSUB-Kommando. (3)
Zufallszahl für die Bewegung auf dem Bildschirm. (4)
Zufallszahlen für arithmetische Probleme (5) in Verbindung mit der MID$-Funktion zur Darstellung eines Zufallszeichens in einer Zeichenkette. (6)

READ/DATA

Das READ-Kommando stellt Daten aus DATA-Anweisungen, wie in Kapitel 1 erklärt, bereit. Es wurde benutzt in Verbindung mit:
Daten in Zeichenketten. (1,3,6)
Positionsdaten für den Bildschirmaufbau. (5)

RESET

löscht ein Lichtrechteck auf dem Bildschirm auf der angegebenen Position. (4)

RETURN

(siehe Unterroutine)

SET

läßt auf einer bestimmten Bildschirmposition ein kleines Rechteck aufleuchten. (4)

Unterroutine

Bezeichnung für einen Programmteil oder -baustein, der durch GOSUB aufgerufen wird. Unterroutinen müssen mit einem RETURN-Kommando abgeschlossen werden. (1 — 6)

Unterroutinen innerhalb von Unterroutinen. (5 — 6)

HOWARD BUDIN

Computer-spiele mit TRS-80

Sie verfügen bereits über Grundkenntnisse der Programmsprache BASIC? Dann können Sie auch für Ihren TRS 80 einige fabelhafte Spiele programmieren!
Mit einfachen, leichtverständlichen Anweisungen werden Sie schrittweise durch geheimnisvolle und abenteuerliche Programme geführt, die Sie mit Hilfe dieses Buches erweitern und verbessern können.
Was Sie daraus machen, liegt bei Ihnen – Ihrer Phantasie sind keine Grenzen gesetzt!



Deutsche Erstveröffentlichung

Computer/ Neue Medien

ISB N 3-404-63087-4 DM +006.80

T 3-28-00

Österreich S 53,–
Niederlande f 9,15
Frankreich FF 25,–
Italien L 6800

